Руководство пользователя FastReport Online Designer

Версия 2024.1

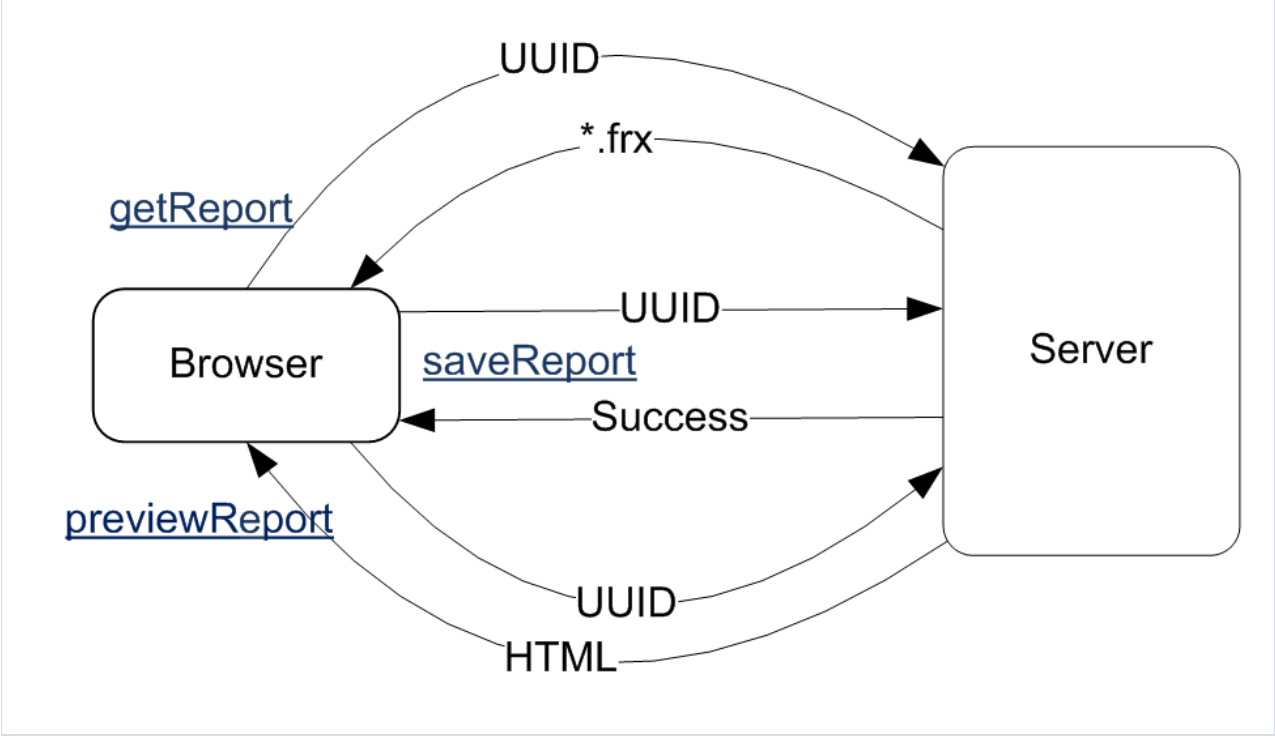
Введение

FastReport Online Designer представляет собой web версию дизайнера отчетов FastReport.Net. Онлайн дизайнер отчетов является RIA (Rich Internet application) приложением, что позволяет запускать его с любого устройства, на котором есть современный интернет-браузер. Онлайн дизайнер будет работать в последних версиях популярных браузеров (Chrome, Firefox, Opera, Safari, IE), в отличии от десктоп версии дизайнера, которая работает только в операционной системе Windows. Но, несмотря на все кроссплатформенные преимущества, онлайн версия уступает десктопной по удобству и функциональности.

Таким образом, FastRport Online Designer позиционируется как редактор .Net отчетов, которые уже были созданы и размещены по какому-либо UUID на сервере. Онлайн дизайнер «общается» с сервером через оговоренный заранее API, в который входит 3 запроса:

- getReport – используется при инициализации. Получает шаблон отчета и передает ее в онлайн дизайнер, который подготавливает отчет к редактированию в браузере.
- previewReport (режим предварительного просмотра) – отредактированный шаблон отчета отправляется на сервер, который строит отчет и возвращает его в html формате. Построение отчета происходит на сервере средствами FastReport.Net.
- saveReport – сохраняет шаблон отчета на сервер.

Для работы каждого из запросов необходимо передавать на сервер UUID отчета с помощью параметра, чтобы идентифицировать отчет на сервере.



Продукт разрабатывается с учетом последних возможностей современных браузеров. Например, благодаря технологии HTML5, единожды загрузив онлайн дизайнер, можно использовать его при отсутствии подключения к сети.

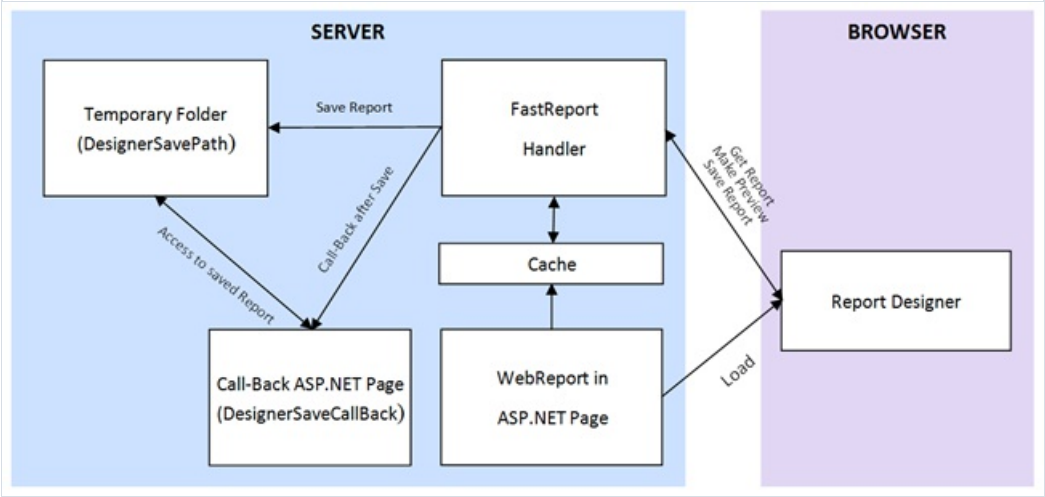
FastReport Online Designer имеет монолитное ядро, к которому подключаются модули, которыми являются компоненты/бэнды/диалоговые компоненты и некоторые другие части системы. Для определения этих модулей и их зависимостей используется технология RequireJS. Такого рода модульность позволяет собирать продукт под нужды клиента исключительно с необходимыми ему компонентами, что позволяет уменьшить размер проекта (так как это web приложение, то размер очень важен и чем он меньше, тем лучше). Для

индивидуальной сборки онлайн дизайнера предусмотрен конструктор дизайнера.

Стоит упомянуть и о других технологиях, применяемых в FastReport Online Designer. Традиционно применяется jQuery, а клиентский движок шаблонов использует jsrender и RequireJS. Для редакторования скрипта отчета используется редактор CodeMirror. Визуально дизайнер в своей основе использует SVG (масштабируемая векторная графика), что идеально подходит для реализации таких компонентов, как LineObject или ShapeObject. Также, SVG позволил реализовать масштабирование страницы отчета.

В текущей версии FastReport Online Designer отсутствует компонент RFID Label. В диалоговых формах, также не все компоненты доступны. В данный момент присутствуют только: Button, CheckBox, CheckedListBox, ComboBox, DateTimePicker, Label, ListBox, MonthCalendar, RadioButton, Text.

Принцип работы



На данный момент возможно использование Online дизайнера совместно с объектами WebReport из продукта FastReport.Net редакций Win+Web, Professional, Enterprise.

После загрузки объекта WebReport на ASP.NET странице выполняется AJAX запрос к handler FastReport для получения содержимого контейнера – в данном случае кода вызова on-line дизайнера в контексте iframe. Код дизайнера отчетов размещается в отдельной папке на сайте приложения. После загрузки дизайнера в браузер пользователя выполняется очередной AJAX запрос к handler из кода дизайнера на получение нужного отчета (getReportByUUIDFrom). На серверной стороне производится подготовка отчета для редактирования и отправка его дизайнеру. В дальнейшем дизайнер может запросить предварительный просмотр отчета. В этом случае текущий отчет отправляется на сервер (makePreviewByUUID), отчет выполняется и результат отправляется обратно в on-line дизайнер, который отображает полученный html в окне предварительного просмотра. Из предварительного просмотра доступна печать отчета, а также экспорт в различные форматы. При завершении редактирования отчета в дизайнере может быть подана команда на сохранение отчета на сервере. В этом случае выполняется AJAX запрос к handler (saveReportByUUIDTo) с содержимым отчета. Сервер производит обработку полученного отчета, сохраняет его во временную папку и выполняет вызов call-back страницы приложения (соответствующие параметры устанавливаются в свойствах WebReport). Важно помнить, что объект WebReport существует в кэше сервера ограниченное время, после чего удаляется из памяти. Время существования объекта WebReport в минутах определяется свойством CacheDelay и по умолчанию равно 60.

Настройка Online Designer

1. Для начала работы с Online дизайнером нужно скопировать папку WebReportDesigner в расположение вашего web-приложения.

2. Файл web.config должен содержать настройки handler для корректной работы объекта WebReport:

Для IIS6:

<system.web>

...

<addpath="FastReport.Export.axd" verb="*" type="FastReport.Web.Handlers.WebExport"/>

</system.web>

для IIS7:

<system.webServer>

<addname="FastReportHandler" path="FastReport.Export.axd" verb="*"
type="FastReport.Web.Handlers.WebExport"/>

</system.webServer>

3. Далее, необходимо проверить настройки дизайнера отчетов в файле WebReportDesigner/scripts/config-data.js:

```
'getReportByUUIDFrom': '/FastReport.Export.axd?getReport=',
'saveReportByUUIDTo': '/FastReport.Export.axd?putReport=',
'makePreviewByUUID': '/FastReport.Export.axd?makePreview=',
```

Данные параметры должны содержать путь к handler FastReport относительно корня web сайта. Если ваш путь отличается от написанного, то его нужно скорректировать, например:

```
'getReportByUUIDFrom': '/oursite/FastReport.Export.axd?getReport=',
```

4. При использовании WebReport в разметке ASPX, кладем на страницу объект и устанавливаем свойства. При использовании в MVC пишем соответствующий код в контроллере:

4.1. Включаем режим редактирования отчетов:

```
webReport.DesignReport = true;
```

4.2. Устанавливаем уникальное имя объекту WebReport, необходимое для дальнейшей идентификации в call-back странице при сохранении отредактированного отчета:

```
webReport.ID = "MyDesignReport1";
```

4.3. Устанавливаем запрет редактирования скрипта отчета в on-line дизайнере, или, если нужно разрешить редактирование - присваиваем true:

```
webReport.DesignScriptCode = false;
```

4.4. Указываем путь к главному файлу дизайнера отчетов, папка с дизайнером должна быть скопирована в соответствующее место web-приложения:

```
webReport.DesignerPath = "~/WebReportDesigner/index.html";
```

4.5. Указываем путь к call-back странице на сайте, вызов которой будет выполнен после сохранения отчета во временную папку, для MVC путь к View (должен быть создан пустой View для корректной работы контроллера):

```
webReport.DesignerSaveCallBack = "~/Home/SaveDesignedReport";
```

или, например, для ASPX:

```
webReport.DesignerSaveCallBack = "~/DesignerCallBack.aspx";
```

При вызове call-back страницы передаются следующие параметры в GET запросе:

```
reportID="here is webReport.ID"&reportUUID="here is saved report file name"
```

В данном контексте reportID соответствует ID объекта WebReport, a reportUUID соответствует имени файла, который сохранен во временную папку. Разработчик должен выполнить дальнейшие действия по сохранению отчета в исходный файл на диске, базу данных либо в облачное хранилище. Временный файл с именем reportUUID необходимо удалить из временной папки после сохранения. Пример кода call-back страницы будет показан в пункте 5. Возможна настройка работы call-back через POST запрос с передачей файла в теле запроса, об этом будет сказано в п.4.6.

4.6. Указываем путь к временной папке, куда будут сохраняться отредактированные отчеты перед вызовом call-back, у папки должны быть установлены права на запись:

```
webReport.DesignerSavePath = "~/App_Data/DesignedReports";
```

Если установить свойство webReport.DesignerSavePath в пустую строку, то передача файла будет осуществлена через POST запрос.

4.7. Устанавливаем время жизни объекта WebReport на сервере в минутах, по умолчанию время равно 60:

```
webReport.CacheDelay = 120;
```

5. Создаем call-back страницу для обработки запросов сохранения отчетов.

5.1. Если используется разметка ASPX, то в обработчике события Page_Load нужно добавить следующий код:

protected void Page_Load(object sender, EventArgs e)

{

string reportID = Request.QueryString["reportID"];

string reportUUID = Request.QueryString["reportUUID"];

```
// 1. Значение reportID идентифицирует объект, который вызывал дизайнер. Значение соответствует свойству
webReport.ID, которое заполнялось при вызове дизайнера.
// 2. Объединяя путь, который мы заполнили в свойстве webReport.DesignerSavePath при вызове дизайнера, и
полученный reportUUID, мы получаем путь к сохраненному временному файлу.
// 3. Данный файл можно открыть и сохранить в нужное нам место на диске, облаке или в базе данных.
// 4. Временный файл нужно удалить после сохранения.
}
```

5.2. Соответственно в разметке MVC создаем метод в контроллере и пустой View. Код в контроллере:

public ActionResult SaveDesignedReport(string reportID, string reportUUID)

{

```
// 1. Значение reportID идентифицирует объект, который вызывал дизайнер. Значение соответствует свойству
webReport.ID, которое заполнялось при вызове дизайнера.
// 2. Объединяя путь, который мы заполнили в свойстве webReport.DesignerSavePath при вызове дизайнера, и
полученный reportUUID, мы получаем путь к сохраненному временному файлу.
// 3. Данный файл можно открыть и сохранить в нужное нам место на диске, облаке или в базе данных.
// 4. Временный файл нужно удалить после сохранения.
return View();
```

}

Если используется передача файла в обработчик Call-back через POST запрос, то нужно добавить перед методом контроллера строку:

[HttpPost]

public ActionResult SaveDesignedReport(string reportID, string reportUUID)

5.3. Можно установить необходимую локализацию для отображения текстовых элементов интерфейса дизайнера с помощью свойства:

```
webReport.DesignerLocale = "en";
```

(вместо "en" может быть указан любой другой буквенный идентификатор языка, перечень доступных локализаций соответствует файлам в папке дистрибутива дизайнера locales).

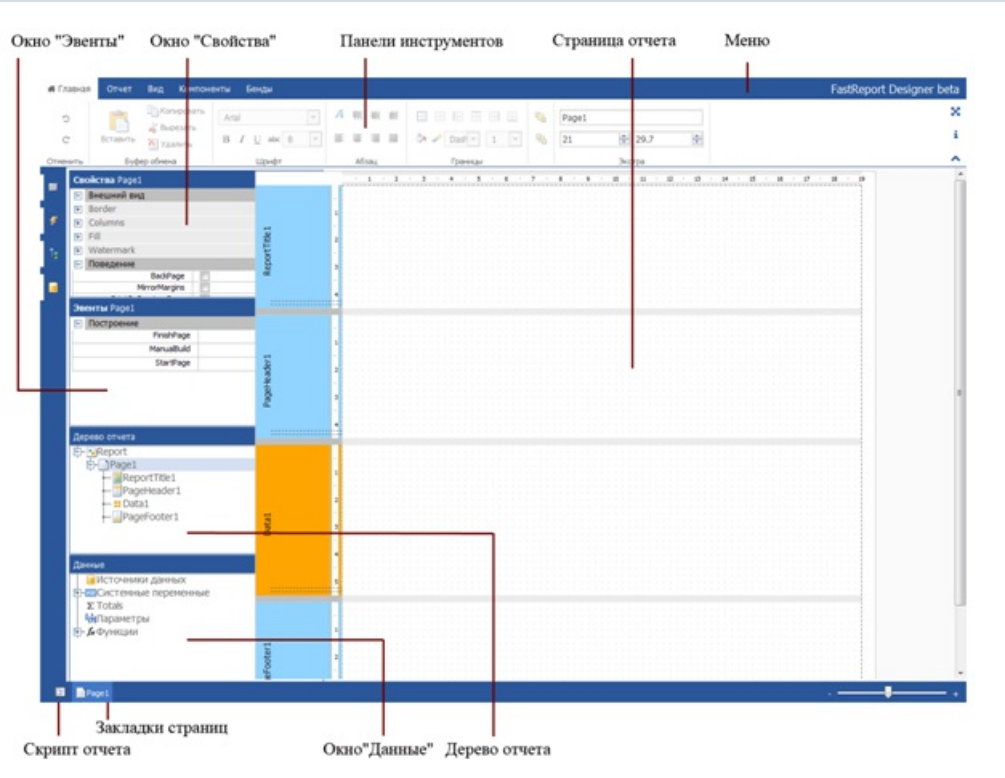
Особое внимание при создании обработчиков Call-back страниц для сохранения отчетов нужно обратить на фильтрацию и проверку получаемых в GET/POST запросе параметров, в частности выполнить их проверку на null.

На странице объект с дизайнером целесообразно разместить в нижней части страницы. Рекомендуемая ширина 100% или не менее 930px. Высоту объекта также рекомендуется установить не менее 600px.

Дизайнер отчетов

Рассмотрим структуру интерфейса дизайнера. Можно выделить следующие области:

- меню;
- страница отчета;
- окно «Свойства»;
- окно «События»;
- дерево отчета;
- окно «Данные»;
- панели инструментов;
- закладки страниц;
- скрипт отчета.



Остановимся на каждом пункте подробнее.

В верхней части дизайнера отчетов располагается главное меню, состоящее из пунктов: Главная, Отчет, Вид, Компоненты, Бэнды. Выбирая пункты, открываем вкладки с панелями инструментов, на манер Microsoft Office 2007.

Панели инструментов вкладки «Главная» предназначены для изменения внешнего вида компонентов отчета.

На вкладке «Отчет» можно сохранить отчет, добавить/удалить страницу отчета, добавить диалог, а также запустить отчет режиме предварительного просмотра.

На вкладке «Вид» задаются настройки сетки страницы отчета. Сетка помогает позиционировать компоненты относительно друг друга.

Вкладка «Компоненты» содержит палитру компонентов FastReport. Компоненты позволяют отображать различные данные в бэндах. Они являются неотъемлемой частью шаблона отчета наряду с бэндами.

Вкладка «Бэнды» содержит палитру бэндов, которые можно добавить в отчет. Бэнды представляют собой

контейнеры для размещения компонентов. Тип бэнда определяет его расположение в отчете.

Страница отчета содержит бэнды и компоненты. Собственно, на странице строится шаблон отчета.

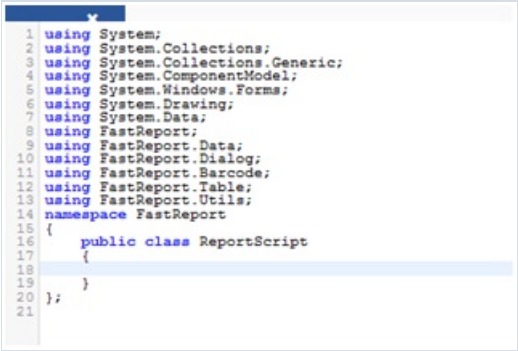
Окно «Свойства», как и другие окна, по умолчанию скрыто. Включить его можно с помощью иконки на боковой панели. Так же можно включить окно «Свойства», дерево отчета и окно «Данные». Для удобства, открытые окна можно переместить в любое место на экране. Чтобы вернуть окно в исходное положение, нужно нажать на значок скрепки в заголовке.

В окне «Свойства» отображаются свойства выбранного объекта отчета. Таким объектом может быть бэнд, компонент или даже страница отчета.

В окне «События» показываются события, доступные для выбранного объекта отчета.

Дерево отчета содержит все объекты отчета в виде иерархического списка. Правым кликом мыши по элементам списка можно вызвать контекстное меню для выбранного объекта.

В самом низу дизайнера отчетов можно увидеть вкладки страниц отчета, а также иконку скрипта отчета. Если перейти в режим скрипта, вместо страницы отчета откроется редактор кода:

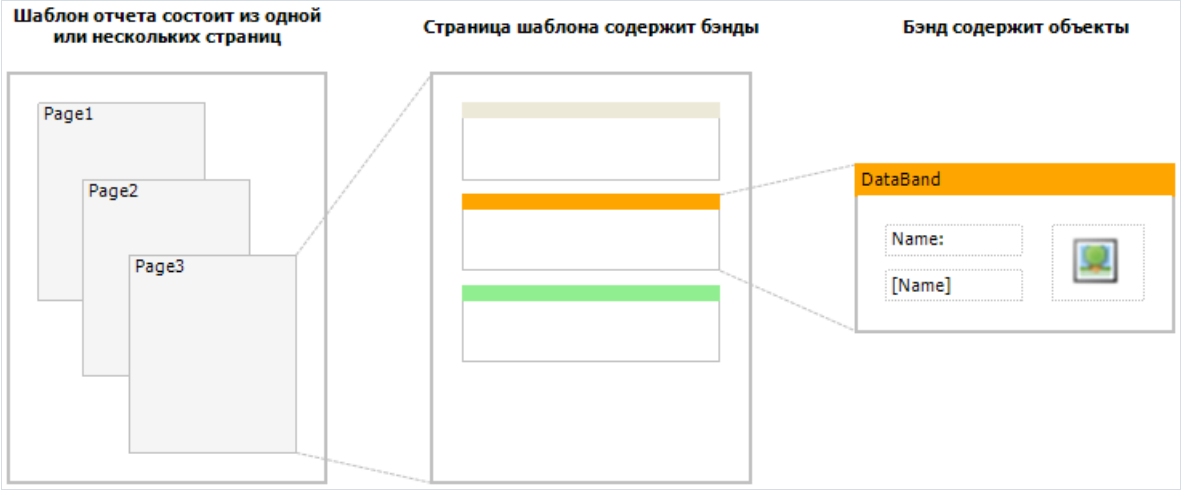


```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Windows.Forms;
using System.Drawing;
using System.Data;
using FastReport;
using FastReport.Data;
using FastReport.Dialog;
using FastReport.Barcode;
using FastReport.Table;
using FastReport.Utils;
namespace FastReport
{
    public class ReportScript
    {

    }
};

```

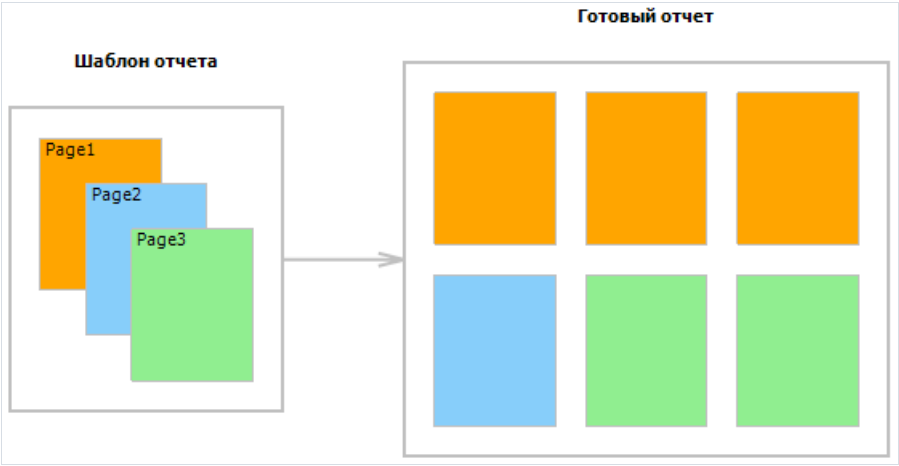
Скрипт отчета позволяет пользователю самостоятельно определять логику работы отчета.

Страницы отчета

Отчет состоит из одной (чаще всего) или нескольких страниц. Страница отчета, в свою очередь, содержит бэнды (англ. band – полоска). На бэндах располагаются объекты отчета, такие как "Текст", "Рисунок" и прочие.

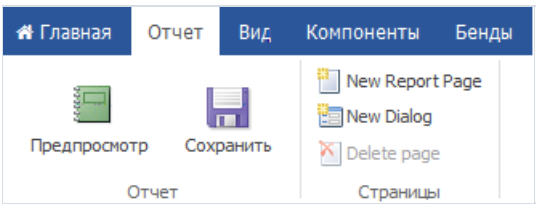


Особенность FastReport – шаблон отчета может состоять из нескольких страниц. Например, вы можете создать шаблон, содержащий титульную страницу и страницу с данными. При построении такого отчета сначала будет напечатана первая страница, затем вторая и т.д. Каждая страница шаблона может генерировать одну или несколько страниц готового отчета – это зависит от содержащихся на ней данных:



Страницы отчета также используются при работе с вложенными отчетами (subreport). В отличие от других генераторов отчетов, вложенные отчеты в FastReport хранятся на отдельной странице шаблона, а не в отдельном файле.

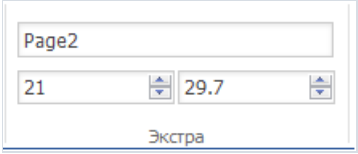
Новый отчет уже содержит одну страницу, но если требуется добавить еще одну, переходим на вкладку «Отчет» и нажимаем на значок новой страницы ("Новая страница").



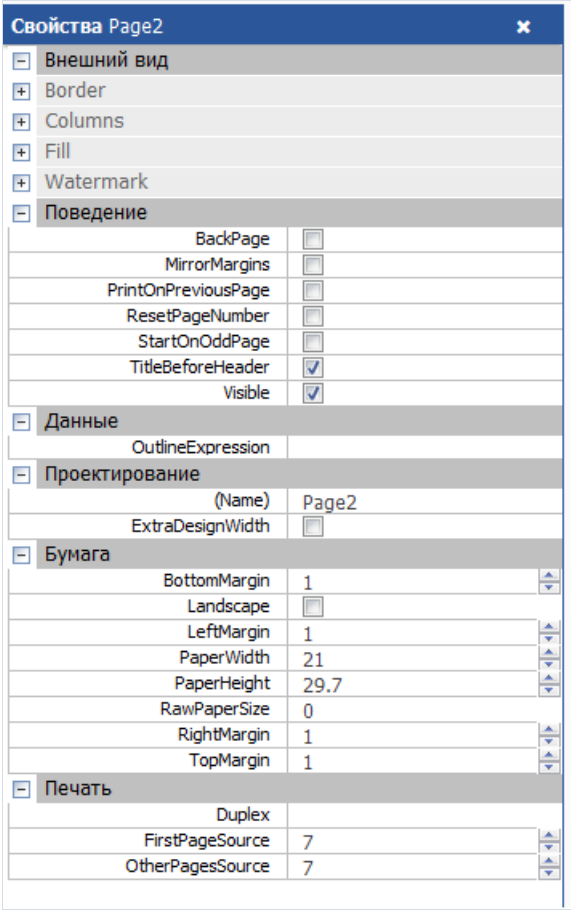
Тут же можно добавить диалоговую форму ("Новый диалог").

Чтобы удалить страницу отчета, необходимо перейти на нужную страницу и нажать на значок удаления ("Удалить страницу"). Если отчет состоит из одной страницы, то значок удаления будет неактивен.

Задать размер страницы можно на вкладке «Главная» в секции «Экстра». Предварительно нужно выбрать страницу из закладок на нижней панели дизайнера.



Другие свойства страницы можно увидеть в окне «Свойства», если выбрать страницу из закладок в нижней части дизайнера.



В разделе «Бумага» помимо размеров страницы, можно задать поля.

Раздел «Печать» позволяет определить двустороннюю печать, источник первой страницы и источник остальных страниц.

Кроме того, доступны настройки колонок страницы, рамок, заливки и др.

Бэнды

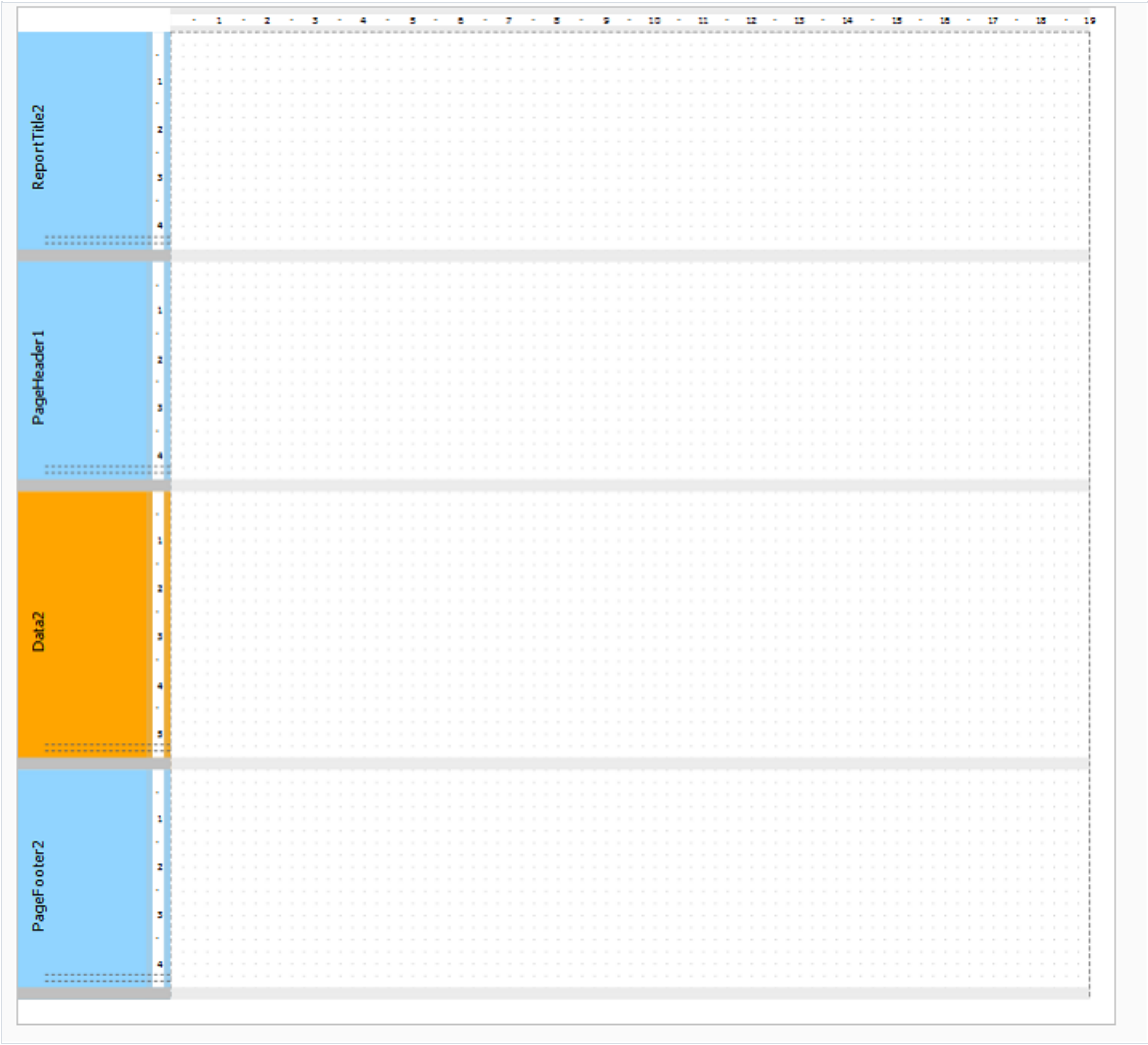
Бэнд (англ. band – полоска) – это объект, который размещается непосредственно на странице отчета и является контейнером для остальных объектов, таких, как "Текст", "Рисунок" и прочих.

Всего в FastReport есть 13 типов бэндов. В зависимости от своего типа, бэнд печатается в определенном месте отчета:

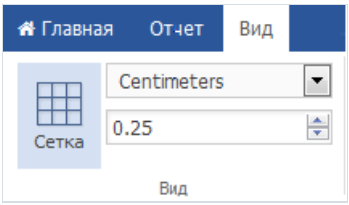
Бэнд	Как печатается
Заголовок отчета	Печатается один раз в самом начале отчета. Вы можете выбрать порядок печати – перед бэндом "Заголовок страницы" или после него – с помощью свойства страницы "TitleBeforeHeader". Изменить это свойство можно с помощью служебного окна "Свойства". По умолчанию свойство равно true, т.е. заголовок отчета печатается перед заголовком страницы.
Подвал отчета	Печатается один раз в конце отчета, после последней строки данных, но перед бэндом "Подвал страницы".
Заголовок страницы	Печатается вверху на каждой странице отчета.
Подвал страницы	Печатается внизу на каждой странице отчета.
Заголовок колонки	Этот бэнд используется при печати многоколоночного отчета (когда в настройках страницы указано количество колонок > 1). Он печатается вверху каждой колонки, после заголовка страницы.
Подвал колонки	Печатается внизу каждой колонки, перед подвалом страницы.
Данные	Этот бэнд подключается к источнику данных и печатается столько раз, сколько строк в источнике.
Заголовок данных	Этот бэнд подключается к бэнду "Данные" и печатается перед первой строкой данных.
Подвал данных	Этот бэнд подключается к бэнду "Данные" и печатается после последней строки данных.
Заголовок группы	Печатается в начале каждой группы, когда значение условия группировки меняется.
Подвал группы	Печатается в конце каждой группы.
Дочерний	Этот бэнд может быть подключен к любому бэнду, в том числе другому дочернему бэнду. Он печатается сразу после своего родителя.
Фоновый	Печатается в виде фона на каждой странице отчета.

Рассмотрим отображение бэндов в дизайнере. Слева на странице отчета размещаются заголовки бэндов. По умолчанию в новый отчет добавляется 4 бэнда:

- ReportTitle (заголовок отчета);
- PageHeader (заголовок страницы);
- Data (данные);
- PageFooter (подвал страницы).



Бэнды могут иметь заливку и рамки, по умолчанию они отключены. Также, бэнды имеют сетку для удобства позиционирования компонентов. Сетку можно настроить в меню «Вид» главного меню.



Изменить размер бэнда можно с помощью мыши. Подведите курсор к нижней части бэнда. Курсор изменит вид. Нажимаем левую кнопку мыши и регулируем высоту бэнда вверх или вниз.

Настройка бендов

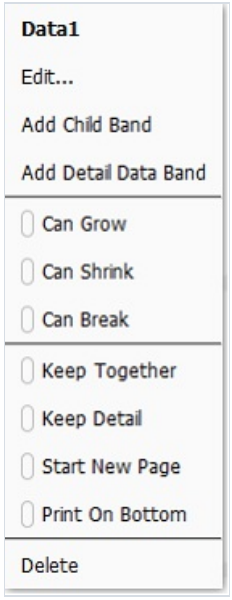
Чтобы добавить бэнд на страницу отчета, нужно перейти на вкладку «Бэнды». Выбираем нужный бэнд и нажимаем на него.

Чтобы добавить «Заголовок данных» или «Подвал данных» нужно предварительно выбрать бэнд Данные на странице отчета.

Чтобы добавить еще один бэнд «Данные», нужно выбрать любой из бэндов на странице отчета кроме уже добавленного бэнда «Данные». Затем добавить бэнд «Данные» с помощью иконки на панели инструментов.

Если выбрать бэнд «Данные» на странице отчета и добавить еще один бэнд «Данные», то будет добавлен детальный бэнд «Данные».

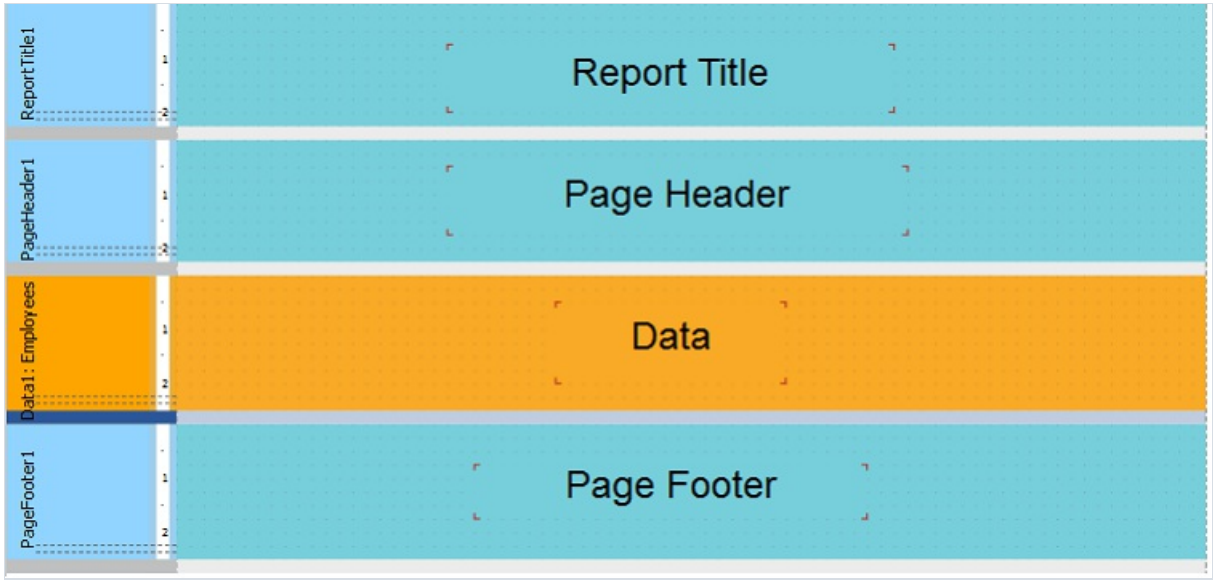
Существует и другой способ добавить детальный бэнд «Данные». С помощью правого клика мышью по бэнду «Данные» на странице отчета вызываем контекстное меню. И выбираем из списка Add Detail Data Band. Из этого же меню можно добавить дочерний бэнд.



Удалить выбранный бэнд можно либо с помощью контекстного меню, либо нажав клавишу Delete. FastReport ограничит ваши действия, которые приведут к созданию некорректного шаблона отчета. Например, если у вас есть бэнд «Заголовок группы», то вы не можете удалить бэнд «Данные» из этой группы. Сначала придется удалить бэнд группировки. Также, при удалении некоторых бэндов, удаляются и связанные с ними бэнды. Например, при удалении бэнда «Данные» его заголовок, подвал, а также дочерний и детальный бэнды тоже будут удалены.

Порядок печати

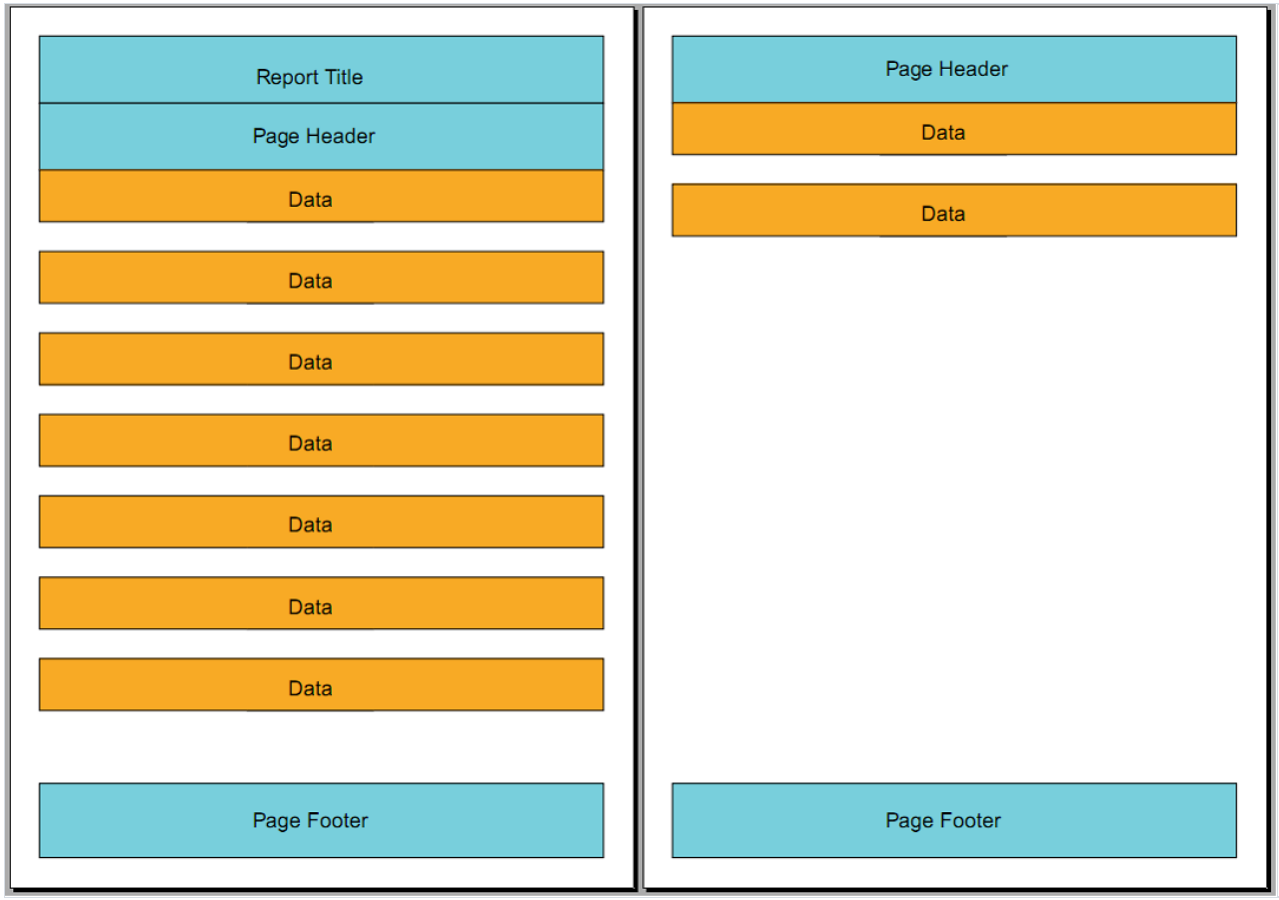
Итак, на странице расположено несколько бэндов. Рассмотрим, как FastReport будет формировать готовый отчет:



Порядок вывода бэндов следующий:

- заголовок отчета;
- заголовок страницы;
- данные. Будет напечатан столько раз, сколько строк в источнике данных, к которому подключен бэнд;
- подвал отчета.

На этом печать отчета заканчивается. Готовый отчет будет выглядеть так:



В процессе печати FastReport проверяет, достаточно ли места на текущей странице готового отчета, чтобы напечатать бэнд. Если места для бэнда нет, алгоритм действий следующий:

- печатается бэнд «Подвал страницы» на текущей странице отчета;
- добавляется новая страница;
- печатается бэнд «Заголовок страницы»;
- продолжается печать бэнда, который не поместился на предыдущей странице.

Свойства бэндов

Все бэнды имеют ряд общих свойств, влияющих на процесс печати. Чтобы просмотреть свойства бэнда, необходимо выбрать его мышью на странице отчета и открыть окно свойств с помощью пиктограммы  на боковой панели.

Свойство	Описание
"Может расти", "Может сжиматься" (CanGrow, CanShrink)	Свойства определяют, может ли бэнд расти или сжиматься в зависимости от размера объектов, которые на нем расположены.
"Может разрываться" (CanBreak)	Если это свойство включено, и на странице недостаточно места для печати бэнда, делается попытка напечатать часть содержимого бэнда на имеющемся месте, т.е. "разорвать" его.
"Формировать новую страницу" (StartNewPage)	Печать бэнда с таким свойством начинается с новой страницы. Обычно это свойство используется при печати групп; при этом каждая группа печатается на новой странице.
"Печатать внизу страницы" (PrintOnBottom)	Бэнд с таким свойством печатается в самом низу страницы, перед бэндом "Подвал страницы". Это может оказаться полезным при печати некоторых документов, где необходимо итоговую сумму печатать внизу страницы.
"Повторять на каждой странице" (RepeatOnEveryPage)	Это свойство имеется у бэндов - заголовков и подвалов данных и групп. Такой бэнд будет напечатан вверху каждой новой страницы, пока идет печать данных.

Компоненты

Компонент «Текст»

Основой представления данных в FastReport служит объект «Текст». В палитре компонентов он выглядит так:

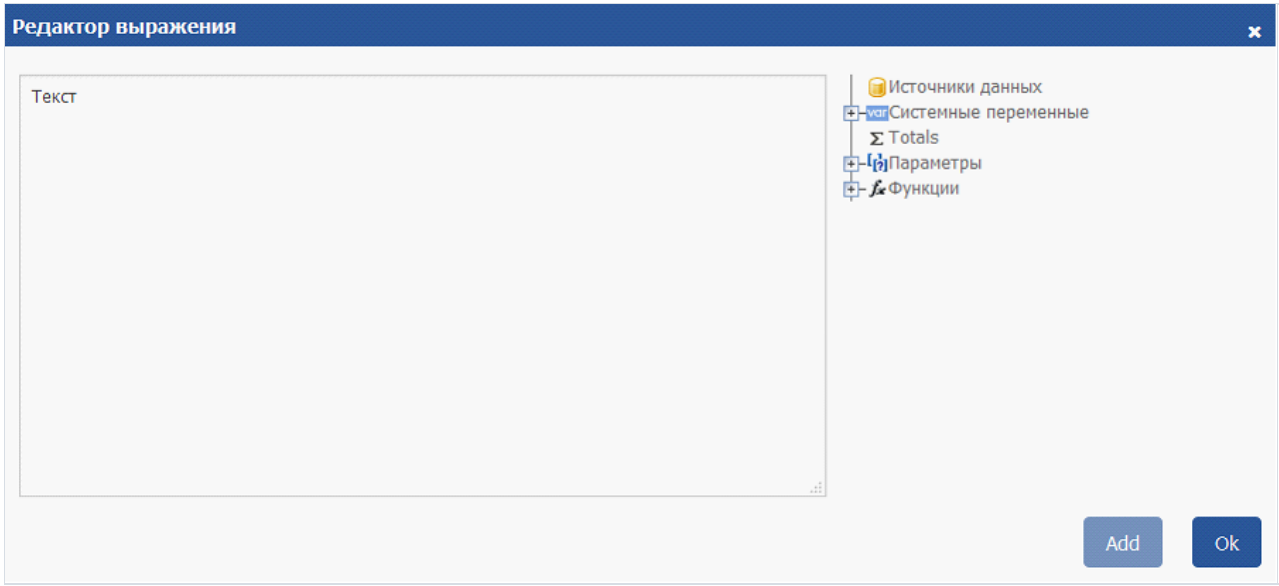
А на странице отчета так:

Объект «Тест» позволяет отображать текстовые данные следующего характера:

- Строки текста;
- Выражения;
- Параметры отчета;
- Итоговые значения;
- Поля из источников данных;
- Системные переменные.

Кроме того, вы можете сочетать эти данные внутри одного текстового объекта.

Чтобы открыть редактор текстового объекта, нужно сделать двойной щелчок мыши по добавленному на страницу отчета объекту. Откроется «Редактор выражения»:



Объект "Текст" может содержать как обычный текст, так и выражения. Причем выражения могут содержаться в объекте вперемешку с текстом. Например:

```
Сегодня [Date]
```

При печати такого объекта все выражения, содержащиеся в тексте, будут вычислены. В текст будет подставлено значение выражения, например:

```
Сегодня 12 сентября 2010г.
```

Как видно, выражения обозначаются с помощью квадратных скобок. Это настраивается в свойстве Brackets, которое по умолчанию содержит строку "[,]". При необходимости вы можете указать другую пару символов, например "<,>" или "<!,!>". В последнем случае выражения в тексте будут выглядеть так:

```
Сегодня <!Date!>
```

Кроме того, можно запретить обработку выражений – за это отвечает свойство AllowExpressions. В этом случае текст будет отображаться "как есть".

В квадратных скобках может содержаться любое выражение, корректное с точки зрения компилятора. Подробнее о выражениях читайте в главе "Выражения".

Например, объект со следующим текстом:

```
2 * 2 = [2 * 2]
```

будет напечатан так:

```
2 * 2 = 4
```

Частая ошибка – попытка написать выражение за пределами квадратных скобок. Напоминаем, что считается выражением и вычисляется только то, что находится в квадратных скобках. Остальной текст печатается "как есть". Например:

```
2 * 2 = [2] * [2]
```

Такой текст будет напечатан следующим образом:

```
2 * 2 = 2 * 2
```

В выражениях могут встречаться элементы, заключенные в собственные квадратные скобки. Например, это обращения к системным переменным. Рассмотрим следующий пример содержимого объекта "Текст":

```
Следующая страница: [[Page] + 1]
```

Текст содержит выражение `[Page] + 1`: это содержимое внешней пары скобок. Page – это системная переменная, которая возвращает номер текущей страницы отчета. Она заключена в собственные скобки. Это обязательно должны быть квадратные скобки, независимо от настройки объекта "Текст" (свойство Brackets, которое мы рассматривали ранее).

Строго говоря, мы должны были использовать две пары квадратных скобок при попытке распечатать

системную переменную Date в ранее рассмотренных примерах:

```
Сегодня [[Date]]
```

Здесь внешняя пара скобок обозначает выражение, внутренняя необходима, так как мы обращаемся к системной переменной. Однако FastReport позволяет опускать лишнюю пару скобок, если в выражении есть только один член.

Для обращения к полям источников данных используется следующая форма записи:

```
[Имя источника.Имя поля]
```

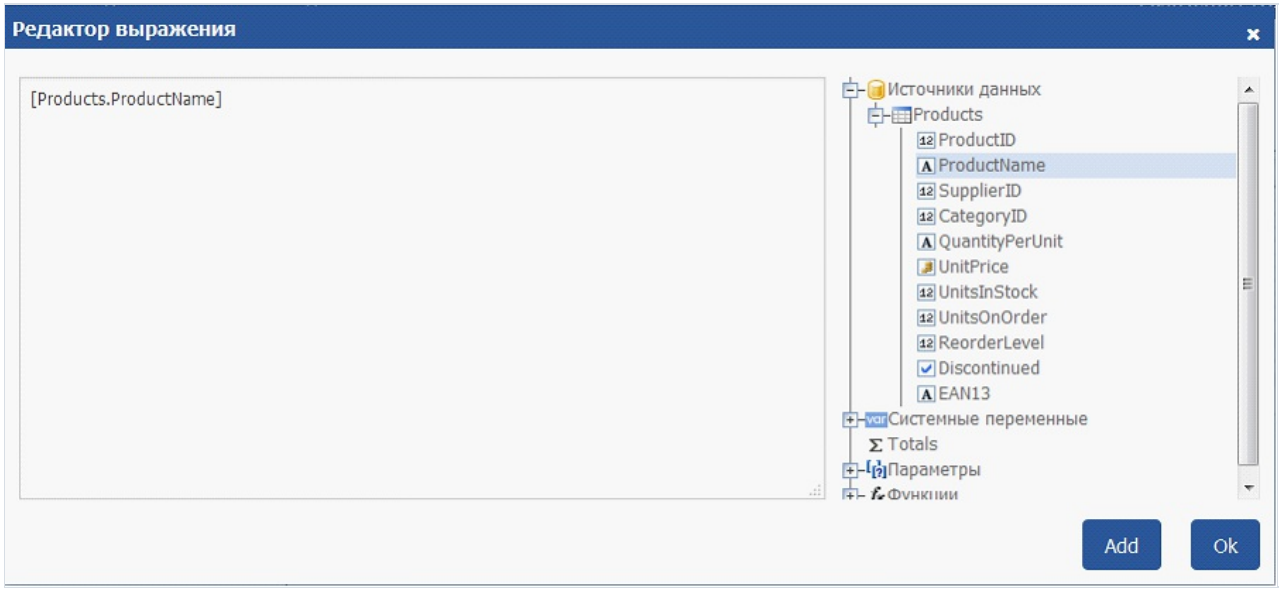
Как видно, здесь так же используются квадратные скобки. Имя источника отделяется от имени поля точкой, например:

```
[Employees.FirstName]
```

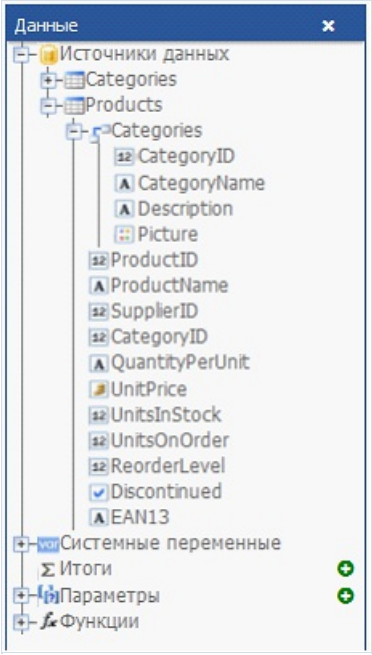
Подробнее об использовании полей в выражениях см. главу "Выражения".

Для вставки поля в объект "Текст" можно использовать несколько способов:

- в редакторе объекта «Текст» вручную вводим название поля;
- в редакторе объекта «Текст» выбираем нужное поле и перетаскиваем его в текст (также можно сделать двойной щелчок на поле):



- Мышью перетаскиваем нужное поле из окна "Данные" на страницу. При этом создается объект "Текст", который содержит ссылку на выбранное поле:



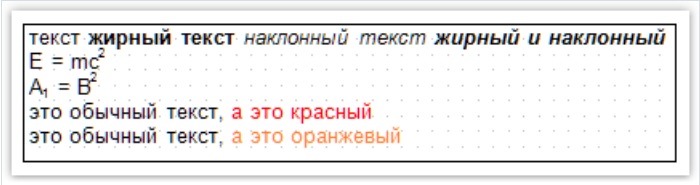
В объекте "Текст" можно использовать некоторые простейшие тэги HTML. По умолчанию тэги отключены; чтобы их включить, в окне "Свойства" установите свойство "HtmlTags" в true. Вот список поддерживаемых тэгов:

Тэг	Описание
...	Жирный текст.
...	Наклонный текст.
<u>...</u>	Подчеркнутый текст.
<strike>... </strike>	Зачеркнутый текст.
_{...}	Подстрочный текст.
^{...}	Надстрочный текст.
**... **	Цвет шрифта. В качестве цвета можно использовать именованный цвет (например, DarkGray), либо шестнадцатиричное значение цвета в формате #RGB, например `#FF8030.`

Продемонстрируем применение тэгов на примерах.

```
текст <b>жирный текст</b> <i>наклонный текст</i> <b><i>жирный и наклонный</b></i>
E = mc<sup>2</sup>
A<sub>1</sub> = B<sup>2</sup>
это обычный текст, <font color=red>а это красный</font>
это обычный текст, <font color=#FF8030>а это оранжевый</font>
```

Этот текст будет отображен следующим образом:



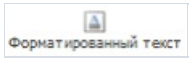
Рассмотрим свойства объекта «Текст»:

Свойство	Описание
"Выражения в тексте" (AllowExpressions)	Это свойство позволяет отключать обработку выражений в тексте объекта. По умолчанию оно включено.
"Поворот" (Angle)	В этом свойстве можно указать угол поворота текста в градусах.
"Автосжатие шрифта" (AutoShrink)	Это свойство позволяет автоматически уменьшить размер шрифта (Font.Size) или его ширину (FontWidthRatio) так, чтобы вместить весь текст.
"Минимальный размер шрифта" (AutoShrinkMinSize)	Это свойство определяет минимальный размер шрифта или минимальное значение свойства FontWidthRatio, в зависимости от значения свойства AutoShrink.
"Автоширина" (AutoWidth)	Это свойство позволяет автоматически подобрать ширину объекта так, чтобы поместить самую длинную строку текста целиком, без переноса слов.
"Скобки" (Brackets)	Это свойство определяет пару символов, которые используются для обозначения выражений в тексте объекта.
"Разрываться в..." (BreakTo)	С помощью этого свойства можно организовать "перетекание" текста между двумя объектами "Текст". Допустим, у нас есть объекты A и B. Объект A содержит большой объем текста, часть которого не помещается в объекте. Если в свойстве BreakTo этого объекта указать объект B, в нем будет напечатан текст, не поместившийся в объекте A.
"Обрезать" (Clip)	Это свойство определяет, надо ли обрезать текст, не поместившийся в объекте. По умолчанию свойство включено.
"Повторяющиеся значения" (Duplicates)	Это свойство определяет, как будут печататься повторяющиеся значения.
"Первая табуляция" (FirstTabOffset)	Это свойство определяет, на сколько пикселов сдвинуть первый символ табуляции.
"Ширина шрифта" (FontWidthRatio)	В этом свойстве можно указать коэффициент масштабирования шрифта по ширине. По умолчанию свойство равно 1. Чтобы увеличить ширину шрифта, укажите значение > 1; чтобы уменьшить ширину, укажите значение между 0 и 1.
"Скрывать значение" (HideValue)	Это строковое свойство позволяет скрывать значения выражений, которые равны заданному значению.
"Скрывать нули" (HideZeros)	Это свойство позволяет скрывать нулевые значения выражений.

Свойство	Описание
"Условное выделение" (Highlight)	Свойство позволяет настроить условное выделение.
"Html тэги" (HtmlTags)	Это свойство позволяет использовать простые html тэги в тексте объекта.
Выравнивание текста (HorzAlign, VertAlign)	Эти свойства задают выравнивание текста внутри объекта по горизонтали и вертикали.
"Межстрочный интервал" (LineHeight)	Свойство позволяет задать межстрочный интервал, в пикселах. Значение по умолчанию = 0, при этом используется стандартный межстрочный интервал.
"Нулевое значение" (NullValue)	Строка, которая будет выводиться вместо null значения. Вам также надо отключить флажок "Преобразовывать null значения" в меню "Отчет/Настройки...".
"Отступы" (Padding)	Свойство позволяет задать отступы текста от краев объекта, в пикселах.
"Справа налево" (RightToLeft)	Это свойство позволяет выводить текст справа налево.
"Ширина табуляции" (TabWidth)	Это свойство позволяет указать ширину символа табуляции, в пикселах.
"Текст" (Text)	Это свойство содержит текст объекта.
"Цвет текста" (TextFill)	Это свойство определяет цвет текста. С помощью редактора этого свойства вы можете выбрать любую из доступных заливок.
"Отсечение" (Trimming)	Это свойство определяет, как показывать текст, который выходит за границы объекта. Это свойство используется только в том случае, если свойство "Перенос слов" отключено.
"Подчеркивание" (Underlines)	Свойство позволяет включить линии подчеркивания под каждой строкой текста. Подчеркивание можно использовать только для текста, выровненного по верхнему краю.
"Перенос слов" (WordWrap)	Это свойство определяет, надо ли переносить текст по словам.
Wysiwyg	Это свойство меняет режим отображения текста таким образом, чтобы добиться максимального соответствия между отображением текста на экране и на распечатке. Этот режим неявно включается, если вы используете выравнивание текста по ширине или нестандартный межстрочный интервал.

Компонент «Форматированный текст»

Объект «Форматированный текст» позволяет отображать многострочный текст в формате RTF с сохранением разметки и стилей. На панели инструментов он выглядит следующим образом:



А на странице отчета он выглядит так же, как и обычный компонент «Текст»:

Важно учитывать, что при экспорте в другие форматы, компонент «Форматированный текст» сохраняется в виде картинки. Поэтому лучше использовать обычный компонент «Текст».

«Форматированный текст» может выводить данные из источника так же, как и обычный объект «Текст». Для этого нужно либо ввести выражение вручную, либо подключить компонент к полю данных с помощью свойства DataColumn.

Объект имеет следующие свойства:

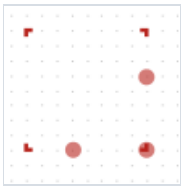
Свойство	Описание
"Выражения в тексте" (AllowExpressions)	Это свойство позволяет отключать обработку выражений в тексте объекта. По умолчанию оно включено.
"Скобки" (Brackets)	Это свойство определяет пару символов, которые используются для обозначения выражений в тексте объекта.
"Поле данных" (DataColumn)	Поле данных, из которого загружать текст объекта.
"Текст" (Text)	Свойства содержит текст объекта в формате RTF.
"Отступы" (Padding)	Свойство позволяет задать отступы текста от краев объекта, в пикселах.

Компонент «Рисунок»

Объект «Рисунок» позволяет размещать изображения в отчете. Поддерживаются следующие графические форматы: BMP, PNG, JPG, GIF, TIFF, ICO, EMF, WMF.

На палитре компонентов объект выглядит так:

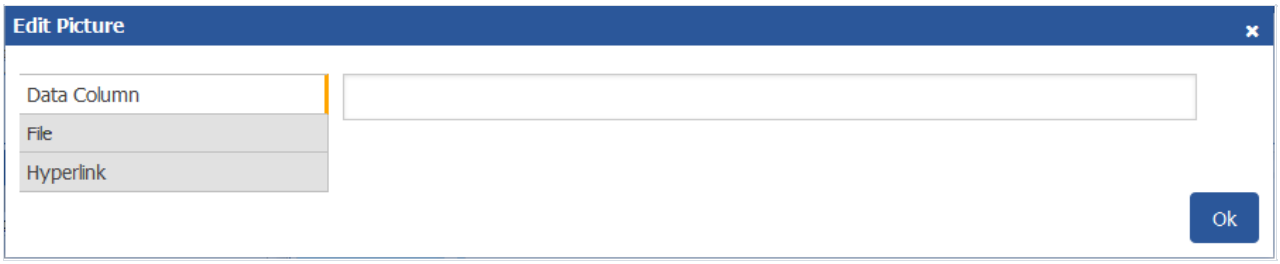
На странице отчета объект отображается так:



Объект может показывать данные из следующих источников:

- Поле из источника данных. Рисунок загружается из поля источника данных. Имя поля хранится в свойстве DataColumn;
- Файл на локальном диске. Рисунок загружается из файла и хранится внутри отчета. Рисунок хранится в свойстве Image;
- Гиперссылка. Рисунок загружается по ссылке каждый раз, когда отчет строится. Внутри отчета рисунок не хранится. Адрес хранится в свойстве ImageLocation. Это может быть как URL, так и ссылка на локальный файл. В последнем варианте вам придется распространять файл рисунка вместе с отчетом.

Чтобы задать источник рисунка нужно открыть его редактор с помощью двойного щелчка мыши:



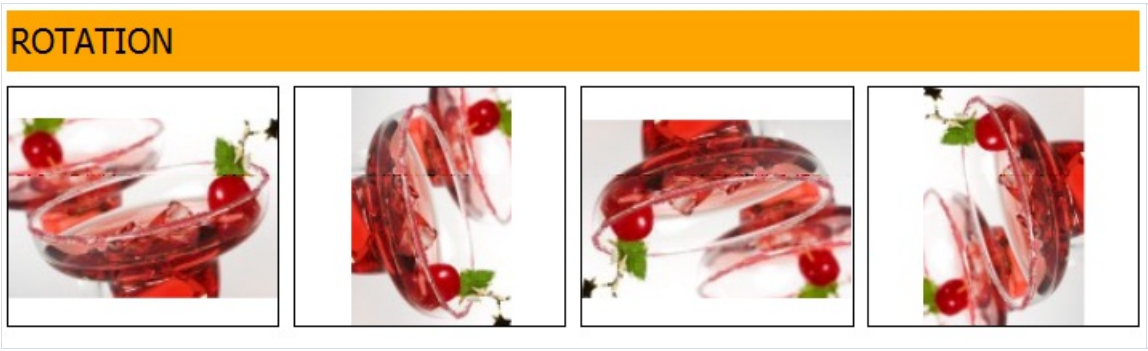
Также вы можете задать источник рисунка в свойствах объекта «Рисунок»: DataColumn, Image, ImageLocation. Если рисунок будет браться из поля источника данных, то можно просто перетащить нужное поле из окна "Данные" на страницу отчета с помощью мыши (drag&drop). При этом будет создан объект "Рисунок", который содержит ссылку на выбранное поле.

В свойстве SizeMode можно настроить режим отображения рисунка:

- Normal (Нормальный). Рисунок выводится в левом верхнем углу объекта без масштабирования;
- StretchImage (Растянуть). Рисунок растягивается до размеров объекта. Пропорции рисунка не соблюдаются;
- AutoSize (Авторазмер). Объект принимает размеры рисунка;
- CenterImage (Центрировать). Рисунок центрируется внутри объекта;
- Zoom (Масштабировать). Рисунок растягивается до размеров объекта с соблюдением пропорций.

Различия между режимами показаны на следующем рисунке:

Есть возможность вращать рисунок с помощью свойства Angle. Результаты поворота рисунка на заданный угол показаны на рисунке:



Объект имеет следующие свойства:

Свойство	Описание
"Угол поворота"(Angle)	Угол, на который нужно повернуть рисунок. Возможные значения для этого свойства - 0, 90, 180, 270.
"Режим отображения" (SizeMode)	Режим отображения рисунка.
"Прозрачность" (Transparency)	Степень прозрачности картинки. Свойство может иметь значение между 0 до 1. Значение 0 (по умолчанию) означает, что картинка непрозрачна.
"Прозрачный цвет" (TransparentColor)	Цвет, который будет прозрачным при отображении картинки.
"Рисунок" (Image)	Собственно рисунок.
"Поле данных" (DataColumn)	Поле данных, из которого загружать рисунок.
"Месторасположение рисунка" (ImageLocation)	Свойство может содержать имя файла или URL. Рисунок будет загружен из указанного места при построении отчета.
"Отступы" (Padding)	Свойство позволяет задать отступы рисунка от краев объекта, в пикселах.
"Показывать ошибку" (ShowErrorImage)	Показывает значок "Нет рисунка" в случае, если рисунок пустой. Это свойство имеет смысл использовать, если рисунок загружается из Интернета.

Компонент «Линия»

Компонент «Линия» может отображать горизонтальную, вертикальную или диагональную линию. Линии могут использоваться для оформления отчета. На панели инструментов объект выглядит так:

На странице отчета он выглядит следующим образом:

Если вам нужна рамка или подчеркивание объекта, лучше использовать свойство «Рамка» объекта. Не стоит перегружать отчет лишними объектами. Это может негативно сказаться при экспорте в некоторые форматы.

Чтобы добавить в отчет линию, нажмите иконку на панели инструментов "Компоненты". Или перетащите компонент за иконку в нужное место на странице отчета.

Чтобы изменить тип линии с горизонтальной на диагональную или вертикальную, включите свойство Diagonal в окне свойств. После этого, линию можно будет поднимать за один край задавая ей любой угол наклона.

Задать стиль отрисовки можно в окне свойств, в разделе Border. Также здесь можно установить цвет и толщину линии.

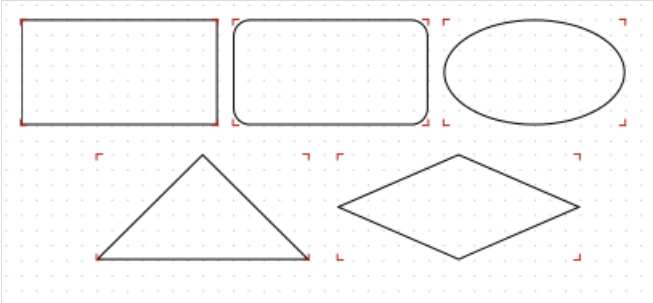
Компонент «Линия» имеет следующие свойства:

Свойство	Описание
"Диагональная" (Diagonal)	Свойство определяет, является ли линия диагональной. Обычную линию можно превратить в диагональную, включив это свойство.
Наконечники линии (StartCap, EndCap)	Эти свойства позволяет задать один из следующих типов наконечников для линии: - эллипс; - прямоугольник; - ромб; - стрелка. Размеры наконечника (ширина и высота) задаются в свойствах Width, Height наконечника. Вы можете настроить наконечник для каждого конца линии.

Компонент «Фигура»

Для оформления отчета могу понадобиться различные фигуры. Объект "Фигура" позволяет добавлять базовые фигуры, изменять их цвет и размер. На палитре компонентов объект выглядит следующим образом:

Доступны следующие фигуры:



- прямоугольник;
- скругленный прямоугольник;
- эллипс;
- треугольник;
- ромб.

Для вставки фигуры в отчет нажмите показанную на рисунке выше иконку на панели инструментов "Компоненты". Затем, в свойстве Shape объекта «Фигура» выберите нужный тип фигуры из списка.

Вы можете делать заливку объекта (свойство Fill) цветом и добавлять рамку (свойство Border).

Объект имеет следующие свойства:

Свойство	Описание
"Тип фигуры" (Shape)	Это свойство позволяет выбрать тип фигуры.
"Закругление" (Curve)	Это свойство позволяет задать закругление для фигуры типа "Скругленный прямоугольник".

Компонент «Флажок»

Объект позволяет отображать в отчете флажок (CheckBox). На палитре компонентов он выглядит следующим образом:

А на странице отчета – так:

Объект может принимать два состояния: true и false. Задать состояние можно следующими способами:

- выбрать значение в свойстве Checked;
- задать поле данных в свойстве DataColumn;
- задать выражение, возвращающее true или false в свойстве Expression;
- перетащить поле из источника данных на страницу отчета. При этом будет добавлен объект «Флажок» с именем поля данных в свойстве DataColumn.

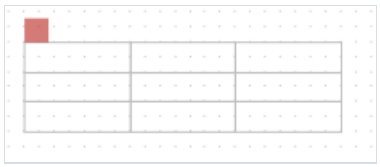
Объект имеет следующие свойства:

Свойство	Описание
Символы флажка (CheckedSymbol, UncheckedSymbol)	Эти свойства определяют тип символа, который отображается во включенном и выключенном состоянии.
"Цвет флажка" (CheckColor)	В этом свойстве можно указать цвет символа флажка.
"Ширина флажка" (CheckWidthRatio)	В этом свойстве можно указать относительную ширину флажка. Ширина флажка зависит от размеров объекта. Вы можете установить любое значение из диапазона 0.2 – 2. По умолчанию свойство равно 1.
"Прятать, если выключен" (HideIfUnchecked)	Свойство позволяет скрыть объект, если он имеет состояние "Выключен".
"Включен" (Checked)	Свойство позволяет напрямую управлять состоянием объекта.
"Поле данных" (DataColumn)	Поле данных, из которого загружать состояние объекта. Поле данных должно быть логического типа (Boolean).
"Выражение" (Expression)	Выражение, которое возвращает состояние объекта. Выражение должно возвращать true или false.

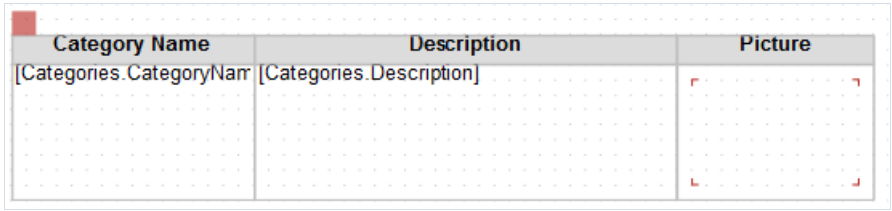
Компонент «Таблица»

Компонент "Таблица" представляет собой упрощенный аналог таблицы Microsoft Excel. На палитре компонентов он выглядит следующим образом:

А на странице отчета – так:



Можно создать статическую таблицу, заполнив ячейки данными вручную. А можно создать динамическую таблицу, используя поля из источника данных. Пример динамической таблицы показан на рисунке:



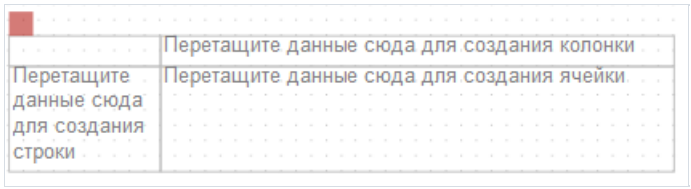
Компонент имеет следующие свойства:

Свойство	Описание
"Количество колонок" (ColumnCount)	Это свойство позволяет быстро создать нужное количество колонок. Если колонок в таблице меньше, они добавляются, если больше – удаляются.
"Количество строк" (RowCount)	Это свойство позволяет быстро создать нужное количество строк. Если строк в таблице меньше, они добавляются, если больше – удаляются.
"Фиксированные колонки" (FixedColumns)	Свойство определяет, сколько колонок в начале таблицы являются фиксированными. Фиксированные колонки образуют заголовок таблицы. Печатью заголовков управляет свойство "Повторять заголовки". Это свойство работает только для таблиц, которые строятся динамически.
"Фиксированные строки" (FixedRows)	Свойство определяет, сколько строк в начале таблицы являются фиксированными. Фиксированные строки образуют заголовок таблицы. Печатью заголовков управляет свойство "Повторять заголовки". Это свойство работает только для таблиц, которые строятся динамически.
"Повторять заголовки" (RepeatHeaders)	Свойство позволяет печатать заголовки таблицы на каждой новой странице. Это свойство работает только для таблиц, которые строятся динамически.

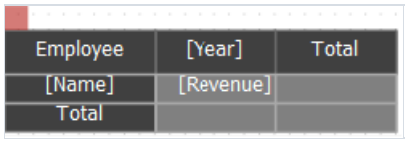
Компонент «Матрица»

Компонент «Матрица» является разновидностью таблицы. При построении матрицы заранее неизвестно, сколько строк и столбцов будет. Это определяется источником данных, к которому она подключена. На палитре компонентов он выглядит следующим образом:

А на странице отчета – так:



Матрицу можно заполнить данными вручную. Но, также, можно создать динамическую матрицу с полями из источника данных:



Компонент имеет следующие свойства:

Свойство	Описание
"Повторять заголовки" (RepeatHeaders)	Если матрица при печати разбивается на несколько страниц, это свойство позволяет печатать заголовки матрицы на каждой новой странице.
"Ячейки одной строкой" (CellsSideBySide)	Свойство определяет, как будут располагаться ячейки матрицы, если матрица имеет несколько значений в ячейках данных. Возможные варианты: - ячейки выводятся рядом (одной строкой); - ячейки выводятся друг под другом.
"Стиль" (Style)	Используя это свойство, можно задать стиль для всей матрицы. Вы можете выбрать один из предустановленных стилей.
"Авторазмер" (AutoSize)	Это свойство, будучи включенным, позволяет подбирать размеры матрицы автоматически. Выключите его, если вы хотите управлять размером объекта вручную.
"Источник данных" (DataSource)	Свойство позволяет подключить матрицу к источнику данных. Это свойство заполняется автоматически при перенесении поля данных в матрицу. Однако, если вы используете выражения в ячейках, проверьте, чтобы это свойство было установлено корректно.
"Фильтр" (Filter)	Это свойство содержит выражение для фильтрации данных, которое будет применено к источнику данных матрицы (см. свойство DataSource).

Компонент «Штрих-код»

Компонент «Штрих-код» позволяет отображать в отчете штрих-код. На палитре компонентов он выглядит так:

А на странице отчета - так:

Объект поддерживает следующие типы штрих-кодов:

Код	Длина	Таблица символов
2 of 5 Interleaved		0-9
2 of 5 Industrial		0-9
2 of 5 Matrix		0-9
Codabar		0-9, -$:/.+
Code128		128 ASCII символов
Code39		0-9,A-Z, -. *$/+%
Code39 Extended		128 ASCII символов
Code93		0-9,A-Z, -. *$/+%
Code93 Extended		128 ASCII символов
EAN8	8	0-9
EAN13	13	0-9
MSI		0-9
PostNet		0-9
UPC A	12	0-9
UPC E0	6	0-9
UPC E1	6	0-9

Код	Длина	Таблица символов
2-Digit Supplement	2	0-9
5-Digit Supplement	5	0-9
PDF417		любые символы
Datamatrix		любые символы
QR Code		любые символы
Aztec Code		любые символы

Данные штрих-кода поступают в объект в виде строки. Строка может содержать любые символы, разрешенные для выбранного типа штрих-кода. Некоторые типы кодов являются цифровыми, остальные могут отображать символьную информацию.

Выбрать тип штрих-кода можно в контекстном меню объекта.

Вы можете подключить объект к данным одним из следующих способов:

- указать строку, содержащую текст объекта, в свойстве Text;
- подключить объект к полю данных с помощью свойства DataColumn;
- указать в свойстве Expression выражение, которое возвращает текст объекта.

Объект имеет следующие свойства:

Свойство	Описание
"Код" (Barcode)	Свойство содержит настройки, специфичные для выбранного типа штрих-кода.
"Поворот" (Angle)	Свойство позволяет задать поворот объекта на один из фиксированных углов – 0, 90, 180, 270 градусов.
"Масштаб" (Zoom)	Свойство задает масштабирование штрих-кода. Это свойство используется только вместе со свойством "Авторазмер".
"Авторазмер" (AutoSize)	Если это свойство включено, объект будет растягиваться, чтобы показать штрих-код целиком. Если свойство отключено, штрих-код будет растянут до размеров объекта.
"Показывать текст" (ShowText)	Свойство определяет, надо ли показывать ли текст в нижней части штрих-кода.
"Поле данных" (DataColumn)	Поле данных, из которого загружать текст объекта.
"Выражение" (Expression)	Выражение, которое возвращает текст объекта.
"Текст" (Text)	Текст объекта.

Свойство	Описание
"Отступы" (Padding)	Свойство позволяет задать отступы от краев объекта, в пикселах.

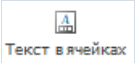
Следующие свойства являются специфичными для выбранного типа штрих-кода. Их можно изменить в окне "Свойства", раскрыв свойство "Barcode" у объекта "Штрих-код":

Свойство	Описание
"Ширина полосок" (WideBarRatio)	Это свойство имеется у всех линейных штрих-кодов. Оно определяет относительный размер широких полосок штрих-кода.
"Контрольная сумма" (CalcCheckSum)	Это свойство имеется у всех линейных штрих-кодов. Оно определяет, надо ли считать контрольную сумму автоматически. Если это свойство отключено, контрольная сумма должна присутствовать в тексте объекта.
"Автокодировка" (AutoEncode)	Это свойство имеется у кода Code128. Этот тип штрих-кода имеет три кодировки – A, B, C. Вы должны либо прямо указать кодировку в тексте, используя управляющие коды, либо включить это свойство, и кодировка будет подобрана автоматически. В тексте можно использовать следующие управляющие коды: &A; START A / CODE A &B; START B / CODE B &C; START C / CODE C &S; SHIFT &1; FNC1 &2; FNC2 &3; FNC3 &4; FNC4 Если вы включили свойство "Автокодировка", все управляющие коды будут игнорированы. Пример использования управляющих кодов в тексте объекта: `&C;1234&B;ABC`
"Отношение сторон" (AspectRatio)	Относится к коду PDF417. Это свойство определяет отношение сторон штрих-кода и используется при автоматическом вычислении размеров (если свойства Columns, Rows не заданы).
"Кодовая страница" (CodePage)	Относится к кодам PDF417 и Datamatrix. Это свойство определяет номер кодовой страницы, которая используется при кодировании символов. Например, для корректной работы с русскими символами задайте значение свойства = 1251.
"Колонки", "Строки"(Columns, Rows)	Относится к коду PDF417. Эти свойства определяет количество колонок и строк в штрих-коде. Если значения свойств равны 0, размер штрих-кода будет подобран автоматически. В этом случае также используется свойство AspectRatio.
"Режим упаковки" (CompactionMode)	Относится к коду PDF417. Это свойство определяет режим упаковки информации.
"Коррекция ошибок" (ErrorCorrection)	Относится к коду PDF417. Это свойство определяет режим коррекции ошибок.
"Размер точки" (PixelSize)	Относится к коду PDF417. Это свойство определяет размер точки штрих-кода, в пикселах. Как правило, высота пиксела должна быть больше его ширины как минимум в 3 раза.

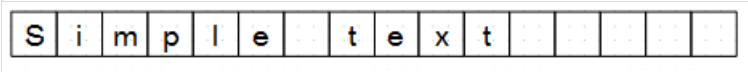
Свойство	Описание
"Кодирование" (Encoding)	Относится к коду Datamatrix. Это свойство определяет тип кодирования информации.
"Размер точки" (PixelSize)	Относится к коду Datamatrix. Это свойство определяет размер точки штрих-кода, в пикселах.
"Размер символа" (SymbolSize)	Относится к коду Datamatrix. Это свойство определяет размер блока штрих-кода.

Компонент «Текст в ячейках»

Компонент «Текст в ячейках» позволяет печатать текст, каждый символ которого выводится в отдельной ячейке. Такой компонент часто используется, например, в анкетах или налоговых формах, когда требуется побуквенно написать ФИО и другие данные. На палитре компонентов он выглядит следующим образом:



А на странице отчета – так:



Вы можете подключить компонент к данным, по аналогии с компонентом «Текст». Для этого в текст компонента поместите ссылку на поле данных, например:

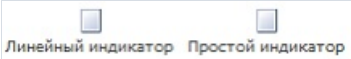
```
[Employees.FirstName]
```

Компонент имеет следующие свойства:

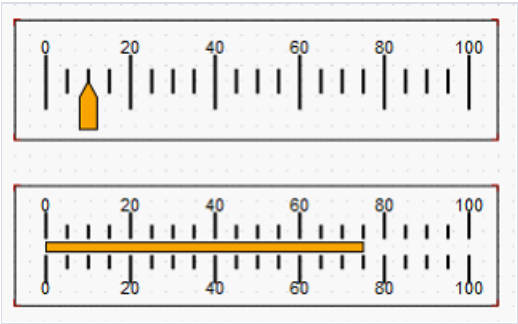
Свойство	Описание
"Размеры ячейки" (CellWidth, CellHeight)	Свойства определяют размеры ячейки. Если оба свойства равны 0 (по умолчанию), размеры ячейки вычисляются автоматически, в зависимости от размера шрифта.
"Промежуток между ячейками"(HorzSpacing, VertSpacing)	Свойства определяют горизонтальный и вертикальный промежуток между соседними ячейками.

Линейный индикатор, простой индикатор

Некоторые отчеты могут потребовать отображения информации в виде индикаторов, как на панелях приборов промышленных объектов. В FastReport есть два компонента для вывода информации в таком виде: линейный индикатор и простой индикатор. На панели инструментов они изображены так:



А на странице отчета – так:



За внешний вид компонентов отвечают свойства Pointer и Scale в окне «Свойства». Вы можете задавать цвет шкал и индикатора, шрифт и толщину линий. Для отображения значения индикатора введите выражение в свойстве Expression. Это выражение может быть полем из источника данных.

Построение отчетов

В этой главе будут описаны способы построения основных типов отчетов. Для построения любого отчета, как правило, нужно выполнить следующие действия:

1. Выбрать или создать данные, которые будут использованы в отчете.
2. Создать структуру отчета, добавив в него необходимые бэнды.
3. Подключить бэнды к источникам данных.
4. Разместить на бэндах объекты "Текст" для печати данных.
5. Задать форматирование у объектов.

Автоматический подбор высоты объектов

Часто приходится печатать текст, размер которого на этапе создания отчета неизвестен. Например, это может быть описание товара. В этом случае приходится решать следующие задачи:

- подобрать высоту объекта так, чтобы он вместил весь текст;
- подобрать высоту бэнда так, чтобы он вместил объекты с переменным количеством текста;
- сдвинуть или изменить высоту других объектов, имеющихся на бэнде, так, чтобы не нарушить общее оформление отчета.

Эти задачи можно решить, используя некоторые свойства объектов и бэндов:

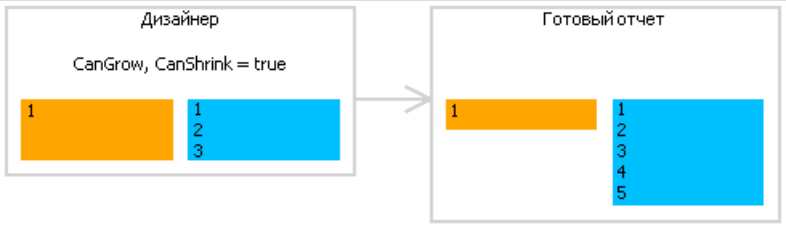
- свойства "Может расти" и "Может сжиматься" позволяют подбирать высоту объекта автоматически;
- свойство "Сдвиг" позволяет сдвигать объекты, расположенные под расширяемыми объектами;
- свойство "Расти вниз" позволяет дотягивать объекты до нижнего края бэнда;
- свойства "Якорь" и "Стыковка" позволяют управлять размером объектов в зависимости от размера бэнда.

Все эти свойства будут рассмотрены ниже.

Свойства "Может расти", "Может сжиматься" (CanGrow, CanShrink)

Эти свойства имеются у бэндов и объектов отчета. Свойства определяют, может ли объект расти или сжиматься в зависимости от размера своего содержимого. Если оба свойства отключены, объект всегда имеет размер, заданный в дизайнере.

Эти свойства очень полезны, если надо напечатать текст, размер которого неизвестен на этапе дизайна. Для того, чтобы объект смог вместить весь текст, у него нужно включить свойства "Может расти"/"Может сжиматься":

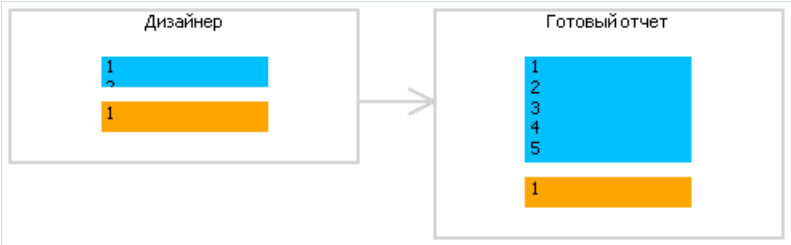


Также эти свойства надо включить у бэнда, чтобы он расширился или сжался в зависимости от размера объектов, которые на нем расположены. Менять размер бэнда могут следующие объекты:

- "Текст";
- "Форматированный текст";
- "Рисунок" (с включенным свойством "Авторазмер");
- "Таблица".

Свойство "Сдвиг" (ShiftMode)

Это свойство имеется у объектов отчета. Его можно изменить в окне "Свойства". Объект с включенным свойством будет сдвинут вниз или вверх, если над ним имеется объект, который расширяется или сжимается:



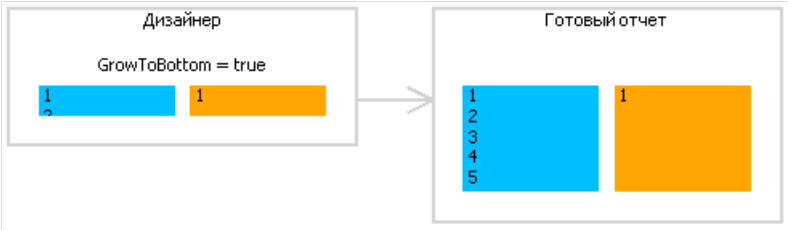
Свойство ShiftMode может иметь одно из значений:

- Всегда (Always). Означает, что объект нужно сдвигать всегда. Это значение по умолчанию.
- Никогда (Never). Означает, что объект сдвигать не нужно.
- При перекрытии (WhenOverlapped). Означает, что объект сдвигать нужно в том случае, если расширяющийся объект находится строго над ним (т.е. оба объекта перекрываются по горизонтали).

Это свойство удобно использовать при печати информации табличного вида, когда несколько ячеек таблицы находятся друг над другом и могут иметь переменное количество текста.

Свойство "Расти вниз" (GrowToBottom)

Это свойство имеется у объектов отчета. Объект с включенным свойством при печати растягивается до нижней границы бэнда, независимо от того, сколько в нем текста:



Это бывает необходимо при печати информации табличного вида. На бэнде может находиться несколько объектов, которые могут растягиваться. Это свойство позволяет "дотягивать" остальные объекты, чтобы не нарушать внешний вид таблицы.

Свойство "Якорь" (Anchor)

Это свойство есть у объектов отчета. Оно определяет, как будет изменяться позиция объекта и/или его размеры при изменении размеров контейнера, на котором он лежит. Используя якорь, можно сделать так, чтобы объект расширялся или сдвигался синхронно с контейнером.

Контейнер, о котором идет речь, в большинстве случаев является бэндом. Но это может быть и объект "Таблица" или "Матрица", которые могут содержать внутри другие объекты.

Свойство может иметь одно из следующих значений, а также любую комбинацию этих значений:

Значение	Описание
Left	Заякорен левый край объекта. При изменении ширины контейнера объект не будет смещаться влево/вправо.
Top	Заякорен верхний край объекта. При изменении высоты контейнера объект не будет смещаться вверх/вниз.
Right	Заякорен правый край объекта. При изменении ширины контейнера расстояние между правыми краями объекта и контейнера будет постоянным. Если при этом заякорен левый край объекта, объект будет расти и сжиматься синхронно с контейнером.
Bottom	Заякорен нижний край объекта. При изменении высоты контейнера расстояние между нижними краями объекта и контейнера будет постоянным. Если при этом заякорен верхний край объекта, объект будет расти и сжиматься синхронно с контейнером.

По умолчанию значение этого свойства равно Left, Top. Это значит, что при изменении размеров контейнера объект меняться не будет. В таблице ниже приведены некоторые часто используемые комбинации значений:

Значение	Описание
Left, Top	Значение по умолчанию. Объект не меняется при изменении размеров контейнера.
Left, Bottom	Объект смещается вверх/вниз при изменении высоты контейнера. Положение объекта относительно нижнего края контейнера остается неизменным.
Left, Top, Bottom	При изменении высоты контейнера высота объекта изменяется синхронно с ним.
Left, Top, Right, Bottom	При изменении ширины и высоты контейнера объект растет/сжимается синхронно с ним.

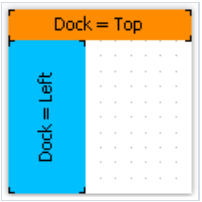
Свойство "Стыковка" (Dock)

Это свойство есть у объектов отчета. Оно определяет, к какой стороне контейнера будет пристыкован объект.

Свойство может иметь одно из следующих значений:

Значение	Описание
None	Значение по умолчанию. Объект не стыкуется.
Left	Объект пристыкован к левому краю контейнера. При этом высота объекта будет равна высоте контейнера*.
Top	Объект пристыкован к верхнему краю контейнера. При этом ширина объекта будет равна ширине контейнера*.
Right	Объект пристыкован к правому краю контейнера. При этом высота объекта будет равна высоте контейнера*.
Bottom	Объект пристыкован к нижнему краю контейнера. При этом ширина объекта будет равна ширине контейнера*.
Fill	Объект занимает все свободное место контейнера.

- это не совсем так, если одновременно стыкуется несколько объектов. На рисунке ниже показано два объекта, первый пристыкован к верхнему краю контейнера, второй – к левому:



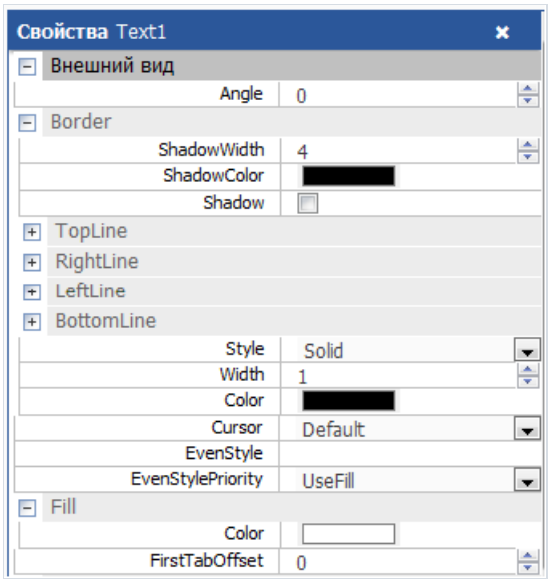
Как видно, высота второго объекта равна высоте свободного места, которое осталось после стыковки первого объекта.

> Стыковка объектов зависит от порядка их создания. Его можно изменить в контекстном меню объекта с помощью пунктов "На передний план" и "На задний план".

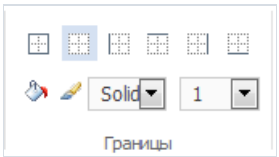
Форматирование объектов

Рамка и заливка

Большинство объектов в FastReport могут иметь рамку и заливку. За это отвечают свойства Border и Fill.



Для добавления рамок компонента нужно использовать панель инструментов «Границы» на вкладке «Главная». Причем цвет, толщина и стиль может быть задан для каждой линии рамки отдельно (в окне свойств).



Как видно из рисунка, можно добавить всю рамку целиком, а можно добавлять рамку отдельными линиями с нужных сторон. Также есть возможность изменить стиль и толщину линий.

Цвет линий рамки задается в окне свойств компонента, в свойстве «Border». Цвет заливки можно изменить, как на панели инструментов «Границы», так и в свойствах компонента.

Формат текста

Панели инструментов «Шрифт» и «Абзац» на вкладке «Главная»:

Здесь есть возможность выбрать и настроить шрифт, изменить положение текста внутри компонента (слева, справа, по центру) по вертикали и горизонтали, задать цвет текста.

Формат данных

Текстовый компонент выводит данные в том формате, в котором они хранятся в источнике данных. Это не всегда удобно. Например, когда дата содержит в себе и время. Чтобы отображалась только дата, необходимо прибегнуть к форматированию данных. Это можно сделать с помощью системной функции String.Format. Строка для отображения текущей даты без времени задается так:

```
Сегодня [String.Format("{0:d}", [Date])]
```

Скрытие значений

Объект "Текст" имеет свойство HideZeros, которое позволяет скрывать нулевые значения. Рассмотрим объект со следующим содержимым:

```
Всего элементов: [CountOfElements]
```

Если значение переменной CountOfElements равно 0, и свойство HideZeros равно true, то объект будет напечатан так:

```
Всего элементов:
```

Объект "Текст" также имеет свойство HideValue строкового типа, которое позволяет скрывать значения выражений, равные заданному значению. Например, если значение свойства равно "0", то будут скрыты все нулевые поля. Это свойство также можно использовать для скрытия "нулевых" дат. Как правило, это даты 1.1.0001 или 1.1.1900. В этом случае значение свойства HideValue должно быть таким:

```
1.1.1900 0:00:00
```

Как вы можете видеть, кроме даты надо написать и время. Это необходимо, потому что значение даты в .Net содержит и время.

> Важное замечание: рассматриваемый механизм зависит от региональных установок, заданных в Панели управления. Это происходит потому, что FastReport сравнивает строки, используя метод ToString() у значения выражения. Этот метод преобразует значение в строку, используя текущие региональные установки. В связи с этим, будьте осторожны при разработке отчетов, которые могут быть запущены на компьютере с другими региональными установками.

Наконец, свойство NullValue объекта "Текст" позволяет выводить какой-либо текст вместо пустых (null) значений. Часто это применяется для того, чтобы напечатать прочерк вместо пустого значения. Рассмотрим объект со следующим содержимым:

```
Всего элементов: [CountOfElements]
```

Если значение переменной CountOfElements равно null, и свойство NullValue равно --, то объект будет напечатан так:

```
Всего элементов: --
```

У объекта "Текст" есть свойство "Повторяющиеся значения" (Duplicates), позволяющее скрывать повторяющиеся значения. Это свойство работает только для объектов "Текст", которые лежат на бэнде "Данные". Повторяющимися считаются одинаковые значения, напечатанные в соседних строках данных.

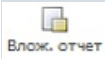
Свойство может иметь одно из следующих значений:

- Показывать (Show) - показывать повторяющиеся значения (значение по умолчанию).
- Прятать (Hide) - прятать объект с повторяющимся значением.
- Очищать (Clear) - очищать текст в объекте, но показывать сам объект.
- Объединять (Merge) - объединять объекты с одинаковыми значениями.

На рисунке продемонстрировано отличие между этими режимами:

Вложенные отчеты

Иногда в определенном месте основного отчета требуется вывести дополнительные данные, которые могут представлять собой отдельный отчет с довольно сложной структурой. Можно попробовать решить эту задачу, используя богатый набор бэндов FastReport. Однако в некоторых случаях предпочтительнее использовать объект "Вложенный отчет".



Объект "Вложенный отчет" представляет собой обычный объект отчета, который можно положить на один из бэндов. При этом FastReport добавит в отчет дополнительную страницу и свяжет ее с объектом вложенного отчета. На этой странице можно создать дополнительный отчет, имеющий любую структуру.

При печати отчета, в котором имеется объект "Вложенный отчет", будет сделано следующее:

1. будет печататься главный отчет, пока не встретится объект "Вложенный отчет";
2. будут печататься бэнды вложенного отчета;
3. будет продолжена печать основного отчета.

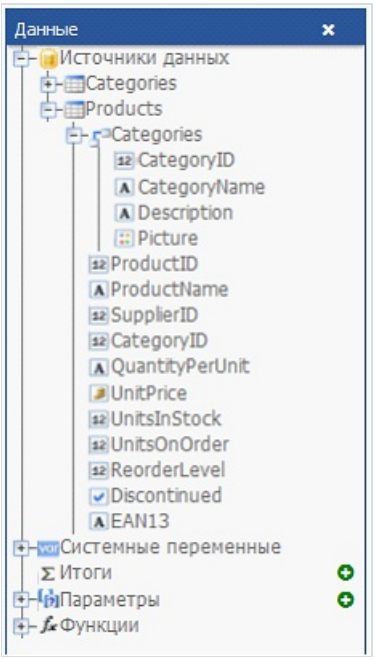
> Поскольку вложенный отчет формируется на листе основного отчета, он не может содержать следующих бэндов: "Заголовок/Подвал отчета", "Заголовок/Подвал/Фон страницы", "Заголовок/Подвал колонки".

Работа с данными

Источник данных

Обратите внимание, что пользователь базы данных, который используется для создания подключения, должен быть ограничен в правах на изменение базы данных, а так же правах на просмотр конфиденциальных данных. В противном случае, вы предоставляете полный доступ к базе данных любому пользователю, который запустит генератор отчетов с этим шаблоном.

В FastReport Online Designer нет возможности добавлять источники данных, но можно создать подключение через мастер подключений, предварительно сконфигурировав необходимый коннектор на сервере. Также, если открыть отчет с добавленным источником данных, он будет показан в окне «Данные». Вы можете перетаскивать поля из источника данных на страницу отчета. При этом будет создаваться текстовый объект с ссылкой на поле данных.



Источник данных может содержать связанные таблицы. Такие таблицы имеют связь по ключевому полю, что позволяет создавать отчеты типа «Главный-подчиненный». Одной записи в главной таблице будет соответствовать одна или несколько записей в подчиненной.

На рисунке показана таблица Products, которая связана с таблицей Categories. FastReport Online Designer не позволяет создавать связи, но допускает их использование при работе с уже созданными отчетами.

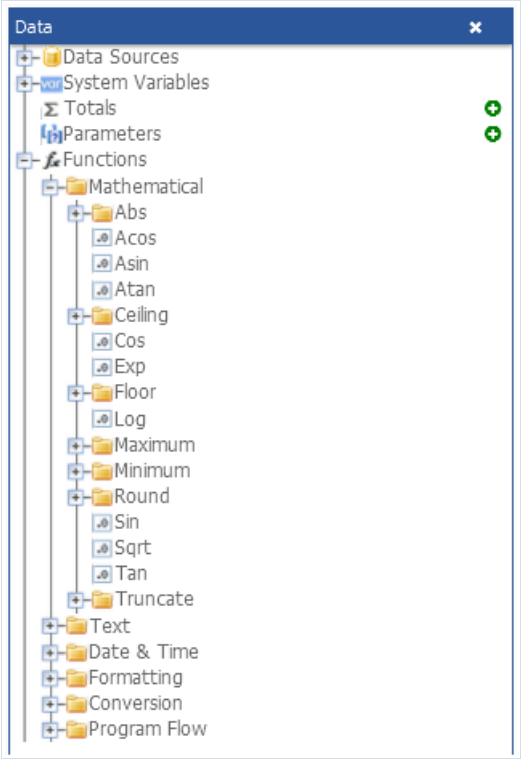
Системные переменные

В FastReport доступно несколько переменных, которые можно использовать в любом месте отчета. Ниже приведен полный список переменных:

Переменная	Описание	
Date	Дата и время старта отчета.	
Page	Номер текущей страницы.	
TotalPages	Общее количество страниц в отчете. Чтобы использовать эту переменную, надо включить двойной проход у отчета. Это можно сделать в меню "Отчет	Свойства...".
PageN	Номер страницы в виде: "Страница N".	
PageNofM	Номер страницы в виде: "Страница N из M".	
Row#	Номер строки данных внутри группы. Это значение сбрасывается при старте новой группы.	
AbsRow#	Абсолютный номер строки данных. Это значение не сбрасывается при старте новой группы.	
Page#	Номер текущей страницы. Если вы объединяете несколько готовых отчетов в пакет, эта переменная вернет номер страницы в пакете.Эта переменная является макросом, т.е. ее значение подставляется в момент отображения компонента в окне просмотра. Использовать ее в выражениях нельзя.	
TotalPages#	Общее количество страниц в отчете. Если вы объединяете несколько готовых отчетов в пакет, эта переменная вернет количество страниц в пакете. Для работы этой переменной двойной проход отчета не требуется.Эта переменная является макросом, т.е. ее значение подставляется в момент отображения компонента в окне просмотра. Использовать ее в выражениях нельзя.	
HierarchyLevel	Текущий уровень иерархии в иерархическом отчете. Верхний уровень имеет значение 1.	
HierarchyRow#	Полный номер строки в иерархическом отчете, имеет вид "1.2.1".	

Функции

FastReport.Net содержит множество встроенных функций (более 60). Все функции разбиты на несколько категорий и доступны через окно «Данные»:



Вы можете использовать функцию в любом выражении, в скрипте (см. главу «Скрипт») или вывести ее значение в объекте «Текст». Например, следующий текст в объекте «Текст»:

```
[Sqrt(4)]
```

будет напечатан как «2» (квадратный корень из 4).

Следующее выражение вернет «4»:

```
Sqrt(4) + 2
```

Рассмотрим способы вставки функции в отчет:

- вы можете перетащить функцию из окна «Данные» на страницу отчета. Будет создан объект «Текст», содержащий вызов функции. Вам придется отредактировать текст, чтобы добавить параметры к вызову функции;
- вы можете перетащить функцию в код скрипта;
- в редакторе выражений вы можете увидеть копию окна «Данные», которое действует аналогично – вы можете перетаскивать элементы из него и помещать их в текст выражения.

Далее мы подробно опишем каждую функцию.

Математические

Abs

Функция	Параметры	Возвращаемое значение
`Abs`	`sbyte value`	`sbyte`
`Abs`	`short value`	`short`
`Abs`	`int value`	`int`
`Abs`	`long value`	`long`
`Abs`	`float value`	`float`
`Abs`	`double value`	`double`
`Abs`	`decimal value`	`decimal`

Возвращает абсолютную величину значения value.

Пример:

```
Abs(-2.2) = 2.2
```

Acos

Функция	Параметры	Возвращаемое значение
`Acos`	`double d`	`double`

Возвращает угол (в радианах), который соответствует косинусу d. Значение d должно быть в диапазоне от -1 до 1.

> Чтобы перевести радианы в градусы, умножьте значение в радианах на 180 / Math.PI.

Пример:

```
Acos(0) * 180 / Math.PI = 90
```

Asin

Функция	Параметры	Возвращаемое значение
`Asin`	`double d`	`double`

Возвращает угол (в радианах), который соответствует синусу d. Значение d должно быть в диапазоне от -1 до 1.

> Чтобы перевести радианы в градусы, умножьте значение в радианах на 180 / Math.PI.

Пример:

```
Asin(0) = 0
```

Atan

Функция	Параметры	Возвращаемое значение
`Atan`	`double d`	`double`

Возвращает угол (в радианах), который соответствует тангенсу d.

> Чтобы перевести радианы в градусы, умножьте значение в радианах на 180 / Math.PI.

Пример:

```
Atan(1) * 180 / Math.PI = 45
```

Ceiling

Функция	Параметры	Возвращаемое значение
`Ceiling`	`double d`	`double`
`Ceiling`	`decimal d`	`decimal`

Возвращает наименьшее целое значение, большее и равное d.

Пример:

```
Ceiling(1.7) = 2
```

Cos

Функция	Параметры	Возвращаемое значение
`Cos`	`double d`	`double`

Возвращает косинус значения d. Значение d должно быть в радианах.

> Чтобы перевести градусы в радианы, умножьте значение в градусах на Math.PI / 180.

Пример:

```
Cos(90 * Math.PI / 180) = 0
```

Exp

Функция	Параметры	Возвращаемое значение
`Exp`	`double d`	`double`

Возвращает число e (2.71828), возведенное в степень d.

Пример:

```
Exp(1) = 2.71828
```

Floor

Функция	Параметры	Возвращаемое значение
`Floor`	`double d`	`double`
`Floor`	`decimal d`	`decimal`

Возвращает наибольшее целое значение, меньшее и равное d.

Пример:

```
Floor(1.7) = 1
```

Log

Функция	Параметры	Возвращаемое значение
`Log`	`double d`	`double`

Возвращает натуральный логарифм значения d.

Пример:

```
Log(2.71828) = 1
```

Maximum

Функция	Параметры	Возвращаемое значение
`Maximum`	`int val1,``int val2`	`int`
`Maximum`	`long val1,``long val2`	`long`
`Maximum`	`float val1,``float val2`	`float`
`Maximum`	`double val1,``double val2`	`double`
`Maximum`	`decimal val1,``decimal val2`	`decimal`

Возвращает максимальное из значений val1, val2.

Пример:

```
Maximum(1,2) = 2
```

Minimum

Функция	Параметры	Возвращаемое значение
`Minimum`	`int val1,``int val2`	`int`
`Minimum`	`long val1,``long val2`	`long`
`Minimum`	`float val1,``float val2`	`float`
`Minimum`	`double val1,``double val2`	`double`
`Minimum`	`decimal val1,``decimal val2`	`decimal`

Возвращает минимальное из значений val1, val2.

Пример:

```
Minimum(1,2) = 1
```

Round

Функция	Параметры	Возвращаемое значение
`Round`	`double d`	`double`
`Round`	`decimal d`	`decimal`

Округляет значение d до ближайшего целого.

Пример:

```
Round(1.47) = 1
```

Функция	Параметры	Возвращаемое значение
`Round`	`double d,``int digits`	`double`
`Round`	`decimal d,``int digits`	`decimal`

Округляет значение d до числа разрядов, указанного в параметре digits.

Пример:

```
Round(1.478, 2) = 1.48
```

Sin

Функция	Параметры	Возвращаемое значение
`Sin`	`double d`	`double`

Возвращает синус значения d. Значение d должно быть указано в радианах.

> Чтобы перевести градусы в радианы, умножьте значение в градусах на Math.PI / 180.

Пример:

```
Sin(90 * Math.PI / 180) = 1
```

Sqrt

Функция	Параметры	Возвращаемое значение
`Sqrt`	`double d`	`double`

Возвращает квадратный корень от значения d.

Пример:

```
Sqrt(4) = 2
```

Tan

Функция	Параметры	Возвращаемое значение
`Tan`	`double d`	`double`

Возвращает тангенс значения d. Значение d должно быть указано в радианах.

> Чтобы перевести градусы в радианы, умножьте значение в градусах на Math.PI / 180.

Пример:

```
Tan(45 * Math.PI / 180) = 1
```

Truncate

Функция	Параметры	Возвращаемое значение
`Truncate`	`double d`	`double`
`Truncate`	`decimal d`	`decimal`

Возвращает целую часть значения d.

Пример:

```
Truncate(1.7) = 1
```

Текстовые

Замечание:

- эти функции не модифицируют переданное в них текстовое значение. Вместо этого они возвращают измененную строку;
- позиция первого символа в строке равна 0. Это следует учитывать при передаче параметров в некоторые функции, например, Insert.

Asc

Функция	Параметры	Возвращаемое значение
`Asc`	`char c`	`int`

Возвращает целое значение, представляющее код символа c.

Пример:

```
Asc('A') = 65
```

Chr

Функция	Параметры	Возвращаемое значение
`Chr`	`int i`	`char`

Возвращает символ с кодом i.

Пример:

```
Chr(65) = 'A'
```

Insert

Функция	Параметры	Возвращаемое значение
`Insert`	`string s, ``int startIndex,``string value`	`string`

Вставляет подстроку value в строку s, начиная с позиции startIndex, и возвращает новую строку.

Пример:

```
Insert("ABC", 1, "12") = "A12BC"
```

Length

Функция	Параметры	Возвращаемое значение
`Length`	`string s`	`int`

Возвращает длину строки s.

Пример:

```
Length("ABC") = 3
```

LowerCase

Функция	Параметры	Возвращаемое значение
`LowerCase`	`string s`	`string`

Переводит все символы строки s в нижний регистр и возвращает результат.

Пример:

```
LowerCase("ABC") = "abc"
```

PadLeft

Функция	Параметры	Возвращаемое значение
`PadLeft`	`string s, ``int totalWidth`	`string`

Дополняет строку s пробелами слева, до длины, указанной в параметре totalWidth, и возвращает новую строку.

Пример:

```
PadLeft("ABC", 5) = "  ABC"
```

Функция	Параметры	Возвращаемое значение
`PadLeft`	`string s, ``int totalWidth,``char paddingChar`	`string`

Дополняет строку s символами paddingChar слева, до длины, указанной в параметре totalWidth, и возвращает новую строку.

Пример:

```
PadLeft("ABC", 5, '0') = "00ABC"
```

PadRight

Функция	Параметры	Возвращаемое значение
`PadRight`	`string s, ``int totalWidth`	`string`

Дополняет строку s пробелами справа, до длины, указанной в параметре totalWidth, и возвращает новую строку.

Пример:

```
PadRight("ABC", 5) = "ABC  "
```

Функция	Параметры	Возвращаемое значение
`PadRight`	`string s, ``int totalWidth,``char paddingChar`	`string`

Дополняет строку s символами paddingChar справа, до длины, указанной в параметре totalWidth, и возвращает новую строку.

Пример:

```
PadRight("ABC", 5, '0') = "ABC00"
```

Remove

Функция	Параметры	Возвращаемое значение
`Remove`	`string s, ``int startIndex`	`string`

Удаляет все символы из строки s, начиная с позиции startIndex, и возвращает новую строку.

Пример:

```
Remove("ABCD", 3) = "ABC"
```

Функция	Параметры	Возвращаемое значение
`Remove`	`string s, ``int startIndex,``int count`	`string`

Удаляет указанное в count количество символов из строки s, начиная с позиции startIndex, и возвращает новую строку.

Пример:

```
Remove("A00BC", 1, 2) = "ABC"
```

Replace

Функция	Параметры	Возвращаемое значение
`Replace`	`string s, ``string oldValue, ``string newValue`	`string`

Заменяет в строке s все вхождения подстроки oldValue на подстроку newValue, и возвращает новую строку.

Пример:

```
Replace("A00", "00", "BC") = "ABC"
```

Substring

Функция	Параметры	Возвращаемое значение
`Substring`	`string s, ``int startIndex`	`string`

Возвращает все символы строки s, начиная с позиции startIndex и до конца строки.

Пример:

```
Substring("ABCDEF", 4) = "EF"
```

Функция	Параметры	Возвращаемое значение
`Substring`	`string s, ``int startIndex,``int length`	`string`

Возвращает количество length символов строки s, начиная с позиции startIndex.

Пример:

```
Substring("ABCDEF", 1, 3) = "BCD"
```

TitleCase

Функция	Параметры	Возвращаемое значение
`TitleCase`	`string s`	`string`

Переводит символы строки s в "титульный" регистр. При этом первый символ каждого слова делается заглавным, остальные символы - строчными.

Пример:

```
TitleCase("john smith") = "John Smith"
```

Trim

Функция	Параметры	Возвращаемое значение
`Trim`	`string s`	`string`

Обрезает незначащие пробелы в начале и конце строки s.

Пример:

```
Trim("  ABC  ") = "ABC"
```

UpperCase

Функция	Параметры	Возвращаемое значение
`UpperCase`	`string s`	`string`

Переводит все символы строки s в верхний регистр и возвращает результат.

Пример:

```
UpperCase("abc") = "ABC"
```

Дата и время

AddDays

Функция	Параметры	Возвращаемое значение
`AddDays`	`DateTime date,``double value`	`DateTime`

Добавляет к дате date количество дней value и возвращает новую дату.

Пример:

```
AddDays(#7/29/2009#, 1) = #7/30/2009#
```

AddHours

Функция	Параметры	Возвращаемое значение
`AddHours`	`DateTime date,``double value`	`DateTime`

Добавляет к дате date количество часов value и возвращает новую дату.

Пример:

```
AddHours(#7/29/2009 1:30#, 1) = #7/29/2009 2:30#
```

AddMinutes

Функция	Параметры	Возвращаемое значение
`AddMinutes`	`DateTime date,``double value`	`DateTime`

Добавляет к дате date количество минут value и возвращает новую дату.

Пример:

```
AddMinutes(#7/29/2009 1:30#, 1) = #7/29/2009 1:31#
```

AddMonths

Функция	Параметры	Возвращаемое значение
`AddMonths`	`DateTime date,``int value`	`DateTime`

Добавляет к дате date количество месяцев value и возвращает новую дату.

Пример:

```
AddMonths(#7/29/2009#, 1) = #8/29/2009#
```

AddSeconds

Функция	Параметры	Возвращаемое значение
`AddSeconds`	`DateTime date,``double value`	`DateTime`

Добавляет к дате date количество секунд value и возвращает новую дату.

Пример:

```
AddSeconds(#7/29/2009 1:30:01#, 1) = #7/29/2009 1:30:02#
```

AddYears

Функция	Параметры	Возвращаемое значение
`AddYears`	`DateTime date,``int value`	`DateTime`

Добавляет к дате date количество лет value и возвращает новую дату.

Пример:

```
AddYears(#7/29/2009#, 1) = #7/29/2010#
```

DateDiff

Функция	Параметры	Возвращаемое значение
`DateDiff`	`DateTime date1,``DateTime date2`	`TimeSpan`

Возвращает интервал - количество дней, часов, минут, секунд между двумя датами.

Пример:

```
DateDiff(#1/2/2009#, #1/1/2009#) = 1.00:00:00
```

DateSerial

Функция	Параметры	Возвращаемое значение
`DateSerial`	`int year,``int month,``int day`	`DateTime`

Создает новое значение DateTime из указанных года (year), месяца (month) и дня (day).

Пример:

```
DateSerial(2009, 7, 29) = #7/29/2009#
```

Day

Функция	Параметры	Возвращаемое значение
`Day`	`DateTime date`	`int`

Извлекает день месяца (1-31) из указанной даты.

Пример:

```
Day(#7/29/2009#) = 29
```

DayOfWeek

Функция	Параметры	Возвращаемое значение
`DayOfWeek`	`DateTime date`	`string`

Возвращает название дня недели (понедельник..воскресенье) указанной даты.

Пример:

```
DayOfWeek(#7/29/2009#) = "среда"
```

DayOfYear

Функция	Параметры	Возвращаемое значение
`DayOfYear`	`DateTime date`	`int`

Возвращает порядковый номер дня в году (1-365) в указанной дате.

Пример:

```
DayOfYear(#7/29/2009#) = 210
```

DaysInMonth

Функция	Параметры	Возвращаемое значение
`DaysInMonth`	`int year,``int month`	`int`

Возвращает количество дней в месяце month указанного года year.

Пример:

```
DaysInMonth(2009, 7) = 31
```

Hour

Функция	Параметры	Возвращаемое значение
`Hour`	`DateTime date`	`int`

Извлекает час (0-23) из указанной даты.

Пример:

```
Hour(#7/29/2009 1:30#) = 1
```

Minute

Функция	Параметры	Возвращаемое значение
`Minute`	`DateTime date`	`int`

Извлекает минуты (0-59) из указанной даты.

Пример:

```
Minute(#7/29/2009 1:30#) = 30
```

Month

Функция	Параметры	Возвращаемое значение
`Month`	`DateTime date`	`int`

Извлекает месяц (1-12) из указанной даты.

Пример:

```
Month(#7/29/2009#) = 7
```

MonthName

Функция	Параметры	Возвращаемое значение
`MonthName`	`int month`	`string`

Возвращает локализованное название месяца (Январь..Декабрь) с номером month.

Пример:

```
MonthName(1) = "Январь"
```

Second

Функция	Параметры	Возвращаемое значение
`Second`	`DateTime date`	`int`

Извлекает секунды (0-59) из указанной даты.

Пример:

```
Second(#7/29/2009 1:30:05#) = 5
```

Year

Функция	Параметры	Возвращаемое значение
`Year`	`DateTime date`	`int`

Извлекает год из указанной даты.

Пример:

```
Year(#7/29/2009#) = 2009
```

Форматирование

Format

Функция	Параметры	Возвращаемое значение
`Format`	`string format,``params object[] args`	`string`

Заменяет каждый элемент формата в указанной строке format значением соответствующего параметра в массиве args.

Например, следующий вызов функции:

```
Format("Name = {0}, hours = {1:hh}", myName, DateTime.Now)
```

содержит элементы формата "{0}" и "{1:hh}". Они заменяются значениями myName и DateTime.Now. Результат может выглядеть так:

```
Name = Alex, hours = 12
```

Элемент формата имеет следующий синтаксис:

```
{index[,alignment][:formatString]}
```

где

- index - обязательный номер значения, которое будет подставлено в это место строки;
- alignment - необязательная ширина форматированного значения;
- formatString - необязательный спецификатор формата.

Для форматирования числовых значений используются следующие стандартные символы-спецификаторы:

Символ	Тип	Описание
C или c	Валюта	Число преобразуется в строку, представляющую денежные единицы. `Format("{0:C}", 10) = "10,00р."`
D или d	Десятичное число	Этот формат доступен только для целых типов. Число преобразуется в строку, состоящую из десятичных цифр (0-9). `Format("{0:D}", 10) = "10"`
E или e	Научный	Число преобразуется в строку вида "-d.ddd…E+ddd" или "-d.ddd…e+ddd", где знак "d" представляет цифру (0-9). `Format("{0:E}", 10) = "1,000000E+001"`
F или f	Фиксированная запятая	Число преобразуется в строку вида "-ddd.ddd…", где знак "d" представляет цифру (0-9). `Format("{0:F}", 10) = "10,00"`

Символ	Тип	Описание
G или g	Общий	Число преобразуется в наиболее короткую запись. `Format("{0:G}", 10) = "10"`
N или n	Число	Число преобразуется в строку вида "-d,ddd,ddd.ddd...", где знак "d" - цифра (0-9), знак "," - разделитель тысяч, а знак "." - разделитель целой и дробной части. `Format("{0:N}", 1234.56) = "1 234,56"`
P или p	Процент	Число преобразуется в строку, представляющую проценты. Преобразуемое число умножается на 100, чтобы соответствовать процентам. `Format("{0:P}", 0.15) = "15,00%"`
X или x	Шестнадцате-ричный	Этот формат доступен только для целых типов. Число преобразуется в строку шестнадцатеричных знаков. Регистр шестнадцатеричных знаков, превосходящих 9, совпадает с регистром указателя формата. `Format("{0:X}", 26) = "1A"`

После символа-спецификатора формата можно указать число. Для чисел с плавающей запятой это будет означать количество знаков после запятой в форматированном значении:

```
Format("{0:C1}", 12.23) = "12,2р."
```

Если стандартных средств форматирования чисел недостаточно, можно использовать символы настраиваемого числового формата:

Символ	Описание
0	Знак-заменитель нуля. Если форматируемое значение содержит цифру в этой позиции, она копируется в выходную строку. В противном случае 0 отображается в выходной строке.
#	Заместитель цифры. Если форматируемое значение содержит цифру в этой позиции, она копируется в выходную строку. В противном случае в выходной строке ничего не записывается.
.	Разделитель дроби. Первый знак "." в строке формата определяет положение разделителя дроби.
,	Разделитель числовых разрядов. Знак "," в строке формата определяет положение разделителя групп разрядов.
%	Заместитель процентов. При использовании знака "%" в строке форматирования число будет умножено на 100, и в соответствующую позицию в выходной строке будет вставлен символ "%".
;	Разделитель секций. Знак ";" служит для разделения секций положительных, отрицательных и нулевых чисел в строке формата.

Примеры использования:

```
Format("{0:$#,##0.00}", 1024.25) = "$1 024,25"
Format("{0:00%}", 0.25) = "25%"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 1024.25) = "$1 024,25"
Format("{0:$#,##0.00;($#,##0.00);Zero}", -1024.25) = "($1 024,25)"
Format("{0:$#,##0.00;($#,##0.00);Zero}", 0) = "Zero"
```

Для форматирования даты/времени используются следующие стандартные символы-спецификаторы:

Символ	Тип	Пример
d	Короткий шаблон даты	"03.08.2009"
D	Полный шаблон даты	"3 августа 2009 г."
f	Полный шаблон даты и времени (короткий шаблон времени)	"3 августа 2009 г. 20:13"
F	Полный шаблон даты и времени (полный шаблон времени)	"3 августа 2009 г. 20:13:33"
g	Общий шаблон даты и времени (короткий шаблон времени)	"03.08.2009 20:13"
G	Общий шаблон даты и времени (полный шаблон времени)	"03.08.2009 20:13:33"
t	Короткий шаблон времени	"20:13"
T	Полный шаблон времени	"20:13:33"

Так же можно использовать символы настраиваемого формата даты/времени:

Символ	Описание
d	Представляет день месяца в виде числа от 1 до 31. Одноразрядные числа форматируются без нуля в начале.
dd	Представляет день месяца в виде числа от 01 до 31.
ddd	Представляет сокращенное название дня недели.
dddd	Представляет полное название дня недели.
f или F	Представляет миллисекунды.
h	Представляет часы числом от 1 до 12. Одноразрядные числа форматируются без нуля в начале.
hh	Представляет часы числом от 01 до 12.
H	Представляет часы числом от 0 до 23. Одноразрядные числа форматируются без нуля в начале.
HH	Представляет часы числом от 00 до 23.
m	Представляет минуты как число от 0 до 59. Одноразрядные числа форматируются без нуля в начале.
mm	Представляет минуты как число от 00 до 59.
M	Представляет месяц как число от 1 до 12. Одноразрядные числа форматируются без нуля в начале.
MM	Представляет месяц как число от 01 до 12.
MMM	Представляет сокращенное название месяца.

Символ	Описание
MMMM	Представляет полное название месяца.
s	Представляет секунды как число от 0 до 59. Одноразрядные числа форматируются без нуля в начале.
ss	Представляет секунды как число от 00 до 59.
y	Представляет год как число из одной или двух цифр.
yy	Представляет год как число из двух цифр.
yyyy	Представляет год как число из четырех цифр.
:	Представляет стандартный разделитель времени.
/	Представляет стандартный разделитель даты.

Примеры использования:

```
Format("{0:d MMM yyyy}", DateTime.Now) = "3 авг 2009"
Format("{0:dd/MM/yyyy}", DateTime.Now) = "03.08.2009"
Format("{0:d MMMM}", DateTime.Now) = "3 августа"
Format("{0:HH:mm}", DateTime.Now) = "23:32"
Format("{0:dd/MM/yyyy HH:mm}", DateTime.Now) = "03.08.2009 23:32"
```

FormatCurrency

Функция	Параметры	Возвращаемое значение
`FormatCurrency`	`object value`	`string`

Форматирует значение value как валюту, используя региональные установки Windows.

Пример:

```
FormatCurrency(1.25) = "1,25р."
```

Функция	Параметры	Возвращаемое значение
`FormatCurrency`	`object value,``int decimalDigits`	`string`

Форматирует значение value как валюту, округляя результат до количества знаков после запятой decimalDigits.

Пример:

```
FormatCurrency(1.25, 1) = "1,3р."
```

FormatDateTime

Функция	Параметры	Возвращаемое значение
`FormatDateTime`	`DateTime value`	`string`

Форматирует значение value как дату/время. Данная функция не возвращает дату или время, если соответствующие части имеют пустые значения. Пустой датой считается дата 1/1/1, пустым временем - 0:00:00.

Пример:

```
FormatDateTime(#1/1/2009#) = "01.01.2009"
FormatDateTime(#1/1/2009 1:30#) = "01.01.2009 1:30:00"
FormatDateTime(#1:30#) = "1:30:00"
```

Функция	Параметры	Возвращаемое значение
`FormatDateTime`	`DateTime value,``string format`	`string`

Форматирует значение value как дату/время, используя именованный формат format. Допустимые значения для параметра format:

"Long Date"

"Short Date"

"Long Time"

"Short Time"

Пример:

```
FormatDateTime(#1/1/2009 1:30#, "Long Date") = "1 января 2009 г."
FormatDateTime(#1/1/2009#, "Short Date") = "01.01.2009"
FormatDateTime(#1:30#, "Short Time") = "01:30"
FormatDateTime(#1:30#, "Long Time") = "1:30:00"
```

FormatNumber

Функция	Параметры	Возвращаемое значение
`FormatNumber`	`object value`	`string`

Форматирует значение value как число, используя региональные установки Windows.

Пример:

```
FormatNumber(1234.56) = "1 234,56"
```

Функция	Параметры	Возвращаемое значение
`FormatNumber`	`object value,``int decimalDigits`	`string`

Форматирует значение value как число, округляя результат до количества знаков после запятой decimalDigits.

Пример:

```
FormatNumber(1234.56, 1) = "1 234,6"
```

FormatPercent

Функция	Параметры	Возвращаемое значение
`FormatPercent`	`object value`	`string`

Форматирует значение value как процент, используя региональные установки Windows.

Пример:

```
FormatPercent(0.15) = "15,00%"
```

Функция	Параметры	Возвращаемое значение
`FormatPercent`	`object value,``int decimalDigits`	`string`

Форматирует значение value как процент, округляя результат до количества знаков после запятой decimalDigits.

Пример:

```
FormatPercent(0.15, 0) = "15%"
```

Конвертирование

ToBoolean

Функция	Параметры	Возвращаемое значение
`ToBoolean`	`object value`	`bool`

Конвертирует значение value в логический тип.

Пример:

```
ToBoolean(1) = true
ToBoolean(0) = false
```

ToByte

Функция	Параметры	Возвращаемое значение
`ToByte`	`object value`	`byte`

Конвертирует значение value в тип byte.

Пример:

```
ToByte("55") = 55
```

ToChar

Функция	Параметры	Возвращаемое значение
`ToChar`	`object value`	`char`

Конвертирует значение value в символ.

Пример:

```
ToChar(65) = 'A'
```

ToDateTime

Функция	Параметры	Возвращаемое значение
`ToDateTime`	`object value`	`DateTime`

Конвертирует значение value в тип DateTime.

Пример:

```
ToDateTime("1/1/2009") = #1/1/2009#
```

ToDecimal

Функция	Параметры	Возвращаемое значение
`ToDecimal`	`object value`	`decimal`

Конвертирует значение value в тип decimal.

Пример:

```
ToDecimal(1) = 1m
ToDecimal("1") = 1m
```

ToDouble

Функция	Параметры	Возвращаемое значение
`ToDouble`	`object value`	`double`

Конвертирует значение value в тип double.

Пример:

```
ToDouble(1) = 1
ToDouble("1") = 1
```

ToInt32

Функция	Параметры	Возвращаемое значение
`ToInt32`	`object value`	`int`

Конвертирует значение value в тип int.

Пример:

```
ToInt32(1f) = 1
ToInt32("1") = 1
```

ToRoman

Функция	Параметры	Возвращаемое значение
`ToRoman`	`object value`	`string`

Конвертирует числовое значение value в римские цифры. value не должно превышать значения 3998.

Пример:

```
ToRoman(9) = "IX"
```

ToSingle

Функция	Параметры	Возвращаемое значение
`ToSingle`	`object value`	`float`

Конвертирует значение value в тип float.

Пример:

```
ToSingle(1m) = 1f
ToSingle("1") = 1f
```

ToString

Функция	Параметры	Возвращаемое значение
`ToString`	`object value`	`string`

Конвертирует значение value в тип string.

Пример:

```
ToString(false) = "False"
ToString(DateTime.Now) = "04.08.2009 13:45"
```

ToWords

Функция	Параметры	Возвращаемое значение
`ToWords`	`object value`	`string`

Конвертирует числовое значение value в сумму прописью на английском.

Пример:

```
ToWords(1024.25) = "One thousand and twenty-four dollars and 25 cents"
```

Функция	Параметры	Возвращаемое значение
`ToWords`	`object value,``string currencyName`	`string`

Конвертирует числовое значение value в сумму прописью на английском. Используется валюта, заданная в параметре currencyName. Возможные значения этого параметра:

"USD"

"EUR"

"GBP"

Пример:

```
ToWords(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

Функция	Параметры	Возвращаемое значение
`ToWords`	`object value,``string one,``string many`	`string`

Конвертирует целочисленное значение value в число прописью на английском. В параметрах one и many задаются словоформы для единственного и множественного числа.

Пример:

```
ToWords(124, "page", "pages") = "One hundred and twenty-four pages"
ToWords(1, "page", "pages") = "One page"
```

ToWordsEnGb

Функция	Параметры	Возвращаемое значение
`ToWordsEnGb`	`object value`	`string`

Конвертирует числовое значение value в сумму прописью на британском английском. Отличие от функции ToWords в следующем:

- по умолчанию используется валюта GBP;
- по-разному переводятся суммы "миллиард" и "триллион".

Пример:

```
ToWordsEnGb(121) = "One hundred and twenty-one pounds and 00 pence"
```

Функция	Параметры	Возвращаемое значение
`ToWordsEnGb`	`object value,``string currencyName`	`string`

Конвертирует числовое значение value в сумму прописью на британском английском. Используется валюта, заданная в параметре currencyName. Возможные значения этого параметра:

"USD"

"EUR"

"GBP"

Пример:

```
ToWordsEnGb(1024.25, "EUR") = "One thousand and twenty-four euros and 25 cents"
```

Функция	Параметры	Возвращаемое значение
`ToWordsEnGb`	`object value,``string one,``string many`	`string`

Конвертирует целочисленное значение value в число прописью на британском английском. В параметрах one и many задаются словоформы для единственного и множественного числа.

Пример:

```
ToWordsEnGb(124, "page", "pages") = "One hundred and twenty-four pages"
ToWordsEnGb(1, "page", "pages") = "One page"
```

ToWordsRu

Функция	Параметры	Возвращаемое значение
`ToWordsRu`	`object value`	`string`

Конвертирует числовое значение value в сумму прописью на русском.

Пример:

```
ToWordsRu(1024.25) = "Одна тысяча двадцать четыре рубля 25 копеек"
```

Функция	Параметры	Возвращаемое значение
`ToWordsRu`	`object value,``string currencyName`	`string`

Конвертирует числовое значение value в сумму прописью на русском. Используется валюта, заданная в параметре currencyName. Возможные значения этого параметра:

"RUR"

"UAH"

"USD"

"EUR"

Пример:

```
ToWordsRu(1024.25, "EUR") = "Одна тысяча двадцать четыре евро 25 евроцентов"
```

Функция	Параметры	Возвращаемое значение
`ToWordsRu`	`object value,``bool male,``string one,``string two,``string many`	`string`

Конвертирует целочисленное значение value в число прописью на русском. В параметре male надо указать true, если существительное - мужского рода. В параметрах one, two и many задаются словоформы для чисел 1, 2 и 5 (1 "лист", 2 "листа", 5 "листов").

Пример:

```
// слово "страница" женского рода - параметр male = false
ToWordsRu(124, false, "страница", "страницы", "страниц") = "Сто двадцать четыре страницы"
// слово "лист" мужского рода - параметр male = true
ToWordsRu(124, true, "лист", "листа", "листов") = "Сто двадцать четыре листа"
```

Условия

Choose

Функция	Параметры	Возвращаемое значение
`Choose`	`double index,``params object[] choice`	`object`

Возвращает элемент массива choice с индексом index. Первый элемент массива имеет индекс = 1.

Пример:

```
Choose(2, "one", "two", "three") = "two"
```

IIf

Функция	Параметры	Возвращаемое значение
`IIf`	`bool expression,``object truePart,``object falsePart`	`object`

Если expression равно true, возвращает значение truePart, иначе возвращает falsePart.

Пример:

```
IIf(2 > 5, "true", "false") = "false"
```

Switch

Функция	Параметры	Возвращаемое значение
`Switch`	`params object[] expressions`	`object`

В параметр expressions передаются пары вида "условие-значение". Возвращает первое значение, условие для которого равно true.

Пример:

```
// вернет одну из строк "a больше 0","a меньше 0","a равно 0" в зависимости от a
Switch(
  a > 0, "a больше 0",
  a < 0, "a меньше 0",
  a == 0, "a равно 0")
```

Итоговые значения

В большинстве отчетов надо выводить некую итоговую информацию: сумма по группе, количество строк в списке и т.п. В FastReport для этих целей существуют так называемые итоговые значения. С их помощью можно подсчитать функцию от определенного значения по диапазону данных.

Для итога нужно указать следующие параметры:

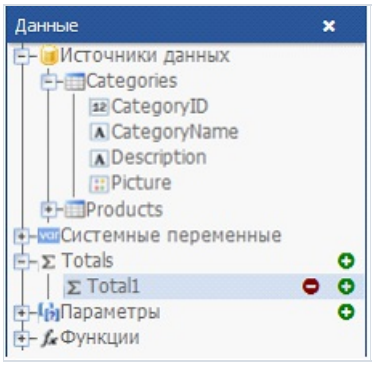
1. тип итоговой функции;
2. диапазон данных, для которого будет вычисляться функция;
3. выражение, которое надо вычислять. Для функции "Количество" выражение не указывается;
4. выражение – условие. Функция будет вычисляться, если это условие выполняется. Этот параметр задавать не обязательно.

Диапазон данных задается следующим образом: указывается бэнд "Данные", для каждой строки которого будет вычисляться функция. Также указывается бэнд, на котором будет печататься итог. После печати этого бэнда значение итога сбрасывается.

Ниже приведен список итоговых функций:

Функция	Описание
Сумма	Вычисляет сумму указанного выражения для диапазона данных.
Минимум	Вычисляет минимальное значение указанного выражения для диапазона данных.
Максимум	Вычисляет максимальное значение указанного выражения для диапазона данных.
Среднее	Находит среднее значение указанного выражения для диапазона данных.
Количество	Возвращает количество строк в диапазоне.

Чтобы добавить итог, необходимо открыть окно «Данные». Напротив узла «Итоги» нажать на кнопку со знаком "Плюс":



Затем, в окне «Свойства» задайте перечисленные выше параметры. После этого созданный итог можно переместить на страницу отчета.

Чтобы удалить итог, используйте кнопку со знаком "Минус" напротив имени итога в окне «Данные».

Параметры отчета

В отчете можно определить параметры. Параметр – это переменная, значение которой может быть определено как в самом отчете, так и вне его (программа, вызывающая отчет, может передавать в него значения параметров – подробнее об этом см. "Руководство программиста"). Параметр может использоваться в выражениях, выводиться в объектах типа "Текст".

Наиболее частые способы использования параметров:

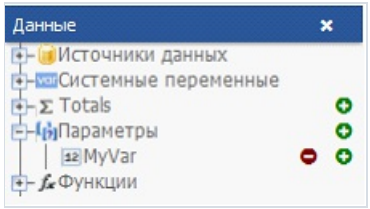
- фильтрация данных по условию, заданному в параметре;
- печать значения параметра в отчете.

Параметр имеет следующие свойства:

Свойство	Описание
Name	Имя параметра может содержать любые символы, кроме точки ".".
DataType	Тип данных параметра.
Expression	Выражение, которое возвращает значение параметра. Подробнее о выражениях смотрите в главе "Выражения". Это выражение будет вычислено при обращении к параметру.
Value	Значение параметра. Это свойство недоступно в дизайнере и может быть заполнено программным способом.

Вы должны настроить свойства Name и DataType. Свойство Expression можно оставить пустым. В этом случае значение параметра надо передавать программным способом.

Чтобы добавить параметр, откройте окно «Данные». Напротив узла «Параметры» нажмите на кнопку со знаком "Плюс". Задайте перечисленные выше свойства. Теперь можно перетащить параметр в нужное место на странице отчета или использовать в фильтрации данных.



Чтобы удалить параметр, используйте кнопку со знаком "Минус" напротив имени параметра в окне «Данные».

Выражения

Во многих местах в FastReport используются выражения. Так, объект "Текст" может содержать выражения, обрамленные квадратными скобками.

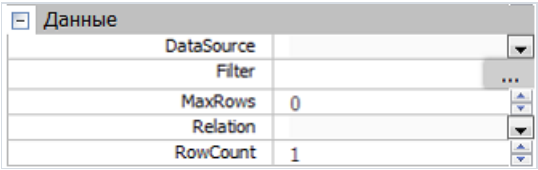
Выражение – это строка кода на языке C# или VB.Net, которая возвращает какое-либо значение. Например:

```
2 + 2
```

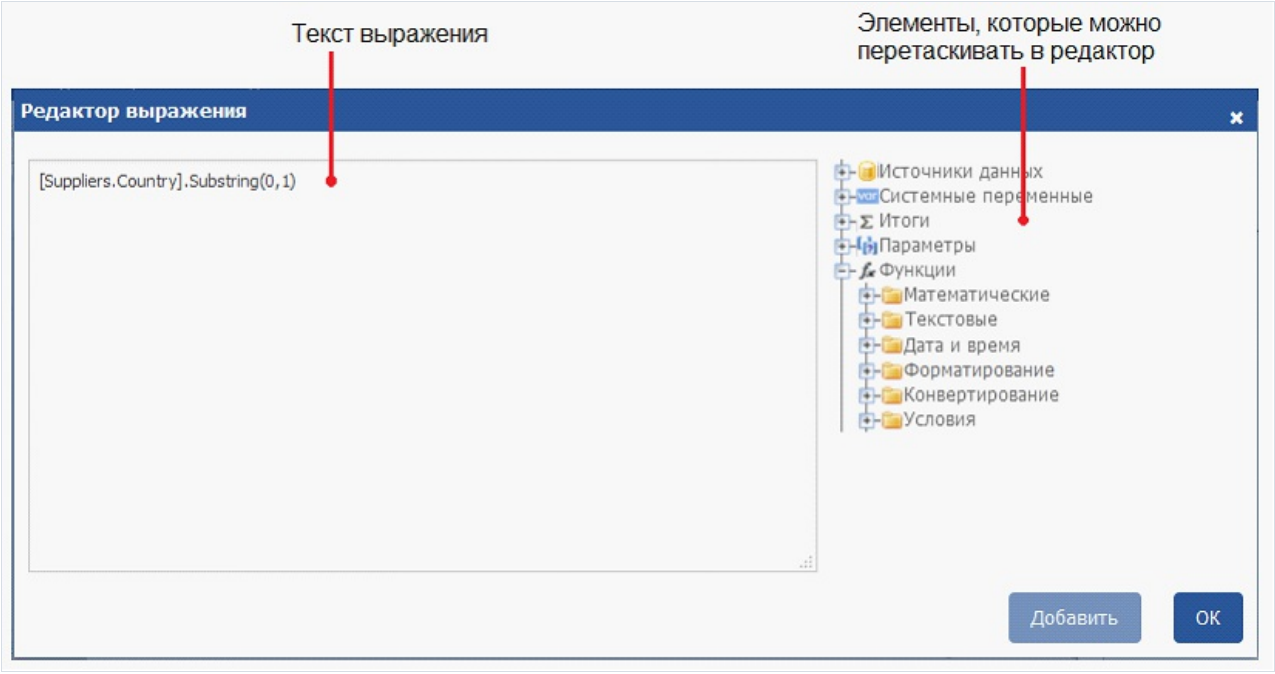
Выражение должно быть написано на том языке, который выбран в качестве скрипта в данном отчете. По умолчанию это C#.

Редактор выражений

Для удобства написания выражения используйте редактор выражений. Его можно вызывать двойным кликом по текстовом объекту или используя кнопку ... в свойствах.



Редактор выражений представляет собой окно, в котором можно ввести текст выражения или вставить в него элементы из окна "Данные":



Обращение к объектам отчета

Для обращения к объектам отчета (например, объекту "Текст") используйте имя объекта. Следующий пример вернет высоту объекта Text1:

```
Text1.Height
```

Для обращения к свойствам отчета используйте переменную Report. Следующий пример вернет имя файла, из которого был загружен отчет:

```
Report.FileName
```

Кроме этого, вы можете обращаться к вложенным свойствам объекта. Следующий пример вернет название отчета:

```
Report.ReportInfo.Name
```

Использование функций .Net

Вы можете использовать любые объекты .Net в выражениях. Следующий пример демонстрирует использование функции Max:

```
Math.Max(5, 10)
```

По умолчанию отчет использует следующие сборки .Net:

```
System.dll
System.Drawing.dll
System.Windows.Forms.dll
System.Data.dll
System.Xml.dll
```

Вам доступны все объекты .Net, объявленные в этих сборках. Если вам нужно получить доступ к другой сборке, добавьте ее название в список сборок отчета. Это можно сделать в десктопной версии дизайнера FastReport.Net.

Например, если вы хотите использовать в отчете функцию, объявленную в вашем приложении, добавьте ссылку на сборку приложения (.exe или .dll) в список сборок отчета. После этого можно обращаться к функции, используя пространство имени (namespace) вашего приложения. Например, если в приложении определена следующая функция:

```
namespace Demo
{
  public static class MyFunctions
  {
    public static string Func1()
    {
      return "Hello!";
    }
  }
}
```

Обратиться к ней в отчете можно так:

```
Demo.MyFunctions.Func1()
```

Если добавить в скрипт директиву "using Demo", это позволит укоротить форму обращения к функции:

```
MyFunctions.Func1()
```

Для обращения к переменным или функциям, которые были вами определены в скрипте отчета, используйте имя переменной или функции:

```
myPrivateVariableThatIHaveDeclaredInScript
MyScriptFunction()
```

> Использовать в выражении можно функции, которые возвращают значение.

Обращение к данным отчета

Помимо стандартных языковых элементов, в выражениях можно использовать следующие элементы отчета:

- поля источников данных;
- системные переменные;
- итоговые значения;
- параметры отчета.

Все эти элементы содержатся в окне "Данные". Любой из этих элементов можно использовать в выражении, заключив его в квадратные скобки. Например:

```
[Page] + 1
```

Это выражение возвращает номер следующей печатаемой страницы. В выражении используется системная переменная Page, которая возвращает номер текущей страницы отчета. Она заключена в квадратные скобки.

Обращение к источникам данных

Для обращения к полям источников данных используется следующая форма записи:

```
[Имя источника.Имя поля]
```

Имя источника отделяется от имени поля точкой, например:

```
[Employees.FirstName]
```

Имя источника может быть составным в случае, если мы обращаемся к источнику данных, используя связь (relation). Например, так можно обратиться к полю связанного источника данных:

```
[Products.Categories.CategoryName]
```

Рассмотрим следующий пример использования полей в выражении:

```
[Employees.FirstName] + “ “ + [Employees.LastName]
```

Здесь надо сделать важное замечание. Каждое поле имеет определенный тип данных - он задается в свойстве DataType поля (его можно увидеть в окне "Свойства", если предварительно выбрать поле данных в окне "Данные"). От того, какой тип имеет поле, зависит, каким образом его можно использовать в выражении. Так, в примере выше, оба поля (имя и фамилия) имеют строковый тип и поэтому их допустимо использовать таким образом. В следующем примере мы попробуем использовать поле Employees.Age числового типа, что приведет к ошибке:

```
[Employees.FirstName] + “ “ + [Employees.Age]
```

Ошибка происходит потому, что нельзя напрямую складывать строку и число. Для этого число надо явным образом преобразовать в строку:

```
[Employees.FirstName] + “ “ + [Employees.Age].ToString()
```

В данном случае мы обращаемся с полем Employees.Age так, будто это целочисленная переменная. Так оно и есть. Мы уже знаем, что все выражения компилируются в исполняемый код. Все нестандартные с точки зрения компилятора вещи (вроде обращения к системным переменным и полям данных) конвертируются в другой вид, понятный компилятору. Так, последнее выражение будет преобразовано в следующий вид:

```
(string)(Report.GetColumnValue("Employees.FirstName")) + " " +
(int)(Report.GetColumnValue("Employees.Age")).ToString()
```

Как видно, FastReport при компиляции выражений заменяет обращения к полям данных следующим образом:

`[Employees.FirstName]` --> `(string)(Report.GetColumnValue("Employees.FirstName"))`

`[Employees.Age]` --> `(int)(Report.GetColumnValue("Employees.Age"))`

То есть, мы можем использовать поле БД в выражениях, как будто это переменная, имеющая определенный тип. Например, следующее выражение вернет первый символ имени сотрудника:

```
[Employees.FirstName].Substring(0, 1)
```

Обращение к системным переменным

В выражениях можно использовать следующие системные переменные (они доступны в окне "Данные"):

Переменная	Тип данных .Net	Описание	
Date	DateTime	Дата и время старта отчета.	
Page	int	Номер текущей страницы.	
TotalPages	int	Общее количество страниц в отчете. Чтобы использовать эту переменную, надо включить двойной проход у отчета. Это можно сделать в меню "Отчет	Свойства...".
PageN	string	Номер страницы в виде: "Страница N".	
PageNofM	string	Номер страницы в виде: "Страница N из M".	
Row#	int	Номер строки данных внутри группы. Это значение сбрасывается при старте новой группы.	
AbsRow#	int	Абсолютный номер строки данных. Это значение не сбрасывается при старте новой группы.	

Каждая переменная имеет определенный тип данных, от этого зависит, как ее можно использовать в выражении. Вот пример выражения, в котором используется дата:

```
[Date].Year
```

Это выражение возвращает текущий год. Так как переменная Date имеет тип DateTime, мы можем обратиться к ее свойству Year. Аналогичным образом можно получить текущий месяц ([Date].Month).

FastReport преобразует обращение к системной переменной в следующий вид (на примере переменной Date):

```
((DateTime)Report.GetVariableValue("Date"))
```

Обращение к итоговым значениям

Для обращения к итоговому значению используйте его имя:

```
[TotalSales]
```

При этом FastReport преобразует обращение к итогу в следующую форму:

```
Report.GetTotalValue("TotalSales")
```

Как видно, здесь тип значения не используется. Это потому, что итоговое значение имеет тип FastReport.Variant. Оно может быть напрямую использовано в любых выражениях, потому что тип FastReport.Variant автоматически приводится к любому типу. Например:

```
[TotalSales] * 0.2f
```

Обращение к параметрам отчета

Для обращения к параметру отчета используйте его имя:

```
[Parameter1]
```

Параметры могут быть вложенными. В этом случае указывается имя родительского параметра и через точку имя дочернего параметра:

```
[ParentParameter.ChildParameter]
```

Параметры, как и поля данных, и системные переменные, имеют определенный тип данных. Он задается в свойстве DataType параметра. От того, какой тип данных имеет параметр, зависит, как его можно использовать в выражении.

FastReport преобразует обращение к параметру отчета в следующий вид (на примере строкового параметра):

```
((string)Report.GetParameterValue("Parameter1"))
```

Скрипт

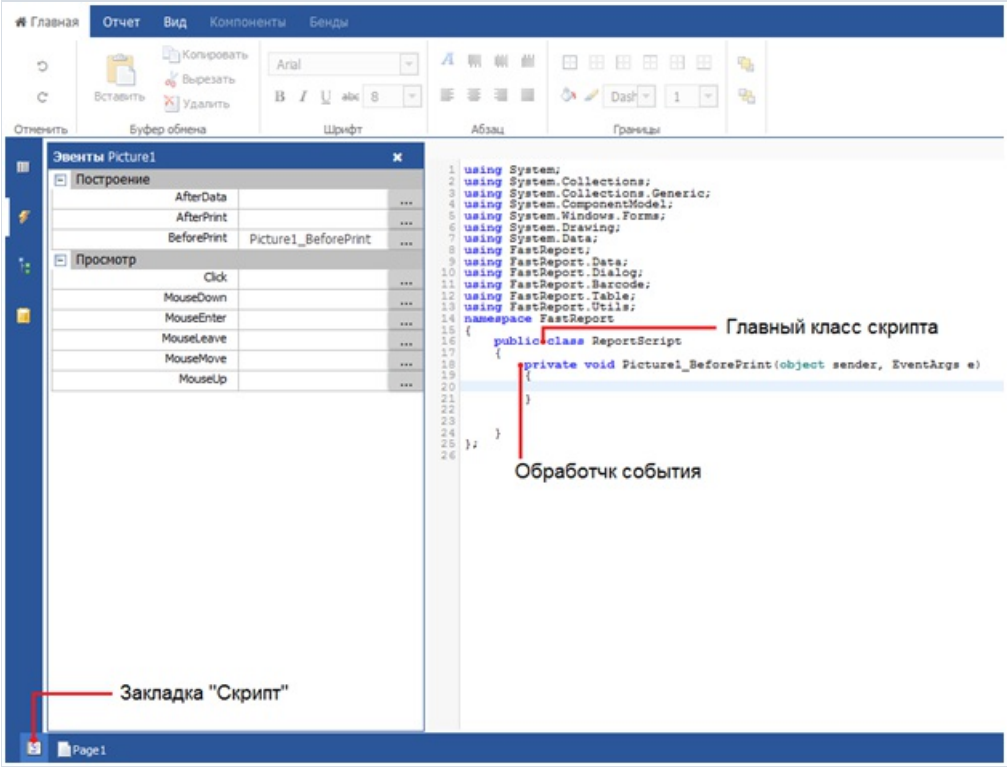
Скрипт – это программа на языке высокого уровня, которая является частью отчета. Скрипт может быть написан на одном из языков .Net:

- C#
- VisualBasic.Net

Область применения скрипта довольно обширна. Используя скрипт, вы можете сделать следующее:

- выполнить обработку данных, которую невозможно сделать штатными средствами ядра FastReport;
- управлять печатью страниц отчета и бэндов на странице;
- управлять взаимодействием элементов управления на диалоговых формах;
- управлять формированием динамических объектов "Таблица";
- и многое другое.

Чтобы увидеть скрипт отчета, переключитесь на закладку "Скрипт" в дизайнере:



В скрипте вы можете:

- добавлять в главный класс скрипта свои переменные, методы, свойства;
- создавать обработчики событий объектов отчета;
- добавлять новые классы в скрипт, если это необходимо. Класс может быть добавлен как перед главным классом ReportScript, так и после него.

Вы не можете:

- удалять, переименовывать или изменять область видимости главного класса ReportScript;
- переименовывать пространство имен, в котором находится главный класс.

При запуске отчета происходит следующее:

- FastReport добавляет в скрипт список переменных, имена которых совпадают с именами объектов

отчета. Это делается перед компиляцией скрипта и позволяет вам обращаться к объектам отчета по их имени;

- в скрипт добавляются выражения, имеющиеся в отчете, в виде функций;
- выполняется компиляция скрипта, если он не пустой;
- инициализируются переменные, которые были неявно добавлены в скрипт;
- обработчики событий, определенные в скрипте, привязываются к объектам отчета;
- запускается отчет.

События

Чтобы максимально гибко управлять отчетом, каждый объект отчета имеет несколько событий, которым можно назначить обработчик – метод из скрипта. Например, в обработчике, привязанном к бэнду "Данные", можно выполнять фильтрацию записей, т.е. скрывать или показывать бэнд в зависимости от каких-либо условий.

Чтобы добавить событие, выберите объект отчета и откройте окно «События». Выберите нужный обработчик и нажмите на иконку ... рядом. Чтобы удалить добавленный обработчик события, удалите его название в окне «События», а также тело обработчика на закладке «Скрипт».

Объект «Отчет» также может иметь события. Чтобы выбрать объект «Отчет», откройте окно «Дерево отчета» и выберите отчет Report1. Теперь, в окне «События» отображаются обработчики событий доступные для отчета.

Рассмотрим процесс формирования отчета и события, которые при этом генерируются. В качестве примера возьмем простой отчет, содержащий одну страницу, один бэнд "Данные" и два объекта "Текст" на бэнде:

В начале отчета вызывается событие StartReport объекта "Отчет". Перед формированием страницы вызывается событие страницы StartPage. Это событие вызывается один раз для каждой страницы шаблона отчета (не путать со страницами готового отчета!). В нашем случае, сколько бы ни было страниц в готовом отчете – событие вызовется один раз, т.к. шаблон отчета состоит из одной страницы.

Далее начинается печать строк бэнда "Данные". Происходит это следующим образом:

1. вызывается событие бэнда BeforePrint;
2. вызываются события BeforePrint всех объектов, лежащих на бэнде;
3. все объекты заполняются данными;
4. вызываются события AfterData всех объектов, лежащих на бэнде;
5. вызывается событие бэнда BeforeLayout;
6. происходит размещение объектов на бэнде (если среди них есть растягиваемые объекты) и подсчет высоты бэнда и его растягивание (если бэнд растягиваемый);
7. вызывается событие бэнда AfterLayout;
8. если бэнд не помещается на свободном месте страницы, формируется новая страница;
9. бэнд и все его объекты выводятся на страницу готового отчета;
10. вызывается событие AfterPrint бэнда;
11. вызывается событие AfterPrint всех объектов бэнда.

Печать строк бэнда происходит до тех пор, пока есть данные в источнике. После этого формирование отчета в нашем случае завершается и вызываются события FinishPage страницы отчета и наконец – событие FinishReport объекта "Отчет".

Таким образом, используя события разных объектов, можно контролировать практически каждый момент формирования отчета. Ключ к правильному использованию событий – полное понимание процесса печати бэндов, изложенного выше в одиннадцати пунктах. Так, большинство действий можно выполнить, используя только событие бэнда BeforePrint – любые изменения, внесенные в объект, будут тут же отображены. Но в этом событии невозможно анализировать, на какой странице будет напечатан бэнд, если он растягиваемый – ведь подсчет высоты бэнда будет выполнен в пункте 6. Это можно сделать с помощью событий AfterLayout в пункте 7 или AfterPrint в пункте 10, но в последнем случае бэнд уже будет напечатан и действия над

объектами ничего не дадут. Одним словом, вы должны четко представлять, в какой момент времени вызывается каждое из событий и использовать те, которые соответствуют поставленной задаче.

Обращение к объектам отчета

Для обращения к объектам отчета (например, объекту "Текст") используйте имя объекта. Следующий пример вернет высоту объекта Text1:

```
float height = Text1.Height;
```

Учтите, что "родными" единицами измерения отчета являются экранные пикселы. Вы должны это помнить при обращении к таким свойствам объектов, как Left, Top, Width, Height. Для перевода пикселов в сантиметры и обратно используйте константы, определенные в классе Units:

```
float heightInPixels = Text1.Height;
float heightInCM = heightInPixels / Units.Centimeters;
Text1.Height = Units.Centimeters * 5; // 5см
```

Объекты Report и Engine

Кроме объектов, которые содержатся в отчете, в скрипте определены две переменные: Report и Engine.

Переменная Report возвращает ссылку на текущий отчет. В таблице ниже приведен список методов объекта Report:

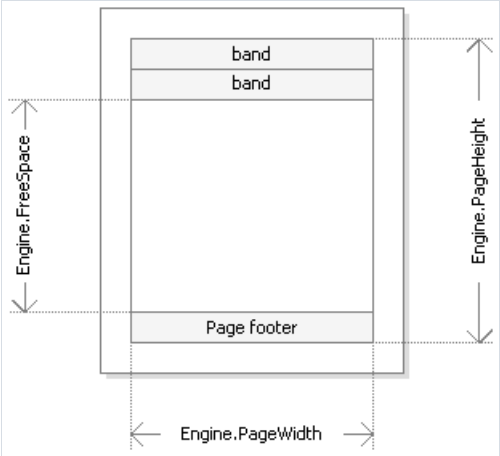
Метод	Описание
`object Calc(`` string expression)`	Вычисляет выражение и возвращает значение. При вызове этого метода первый раз выражение компилируется, что требует некоторого времени.
`object GetColumnValue(`` string complexName)`	Возвращает значение поля источника данных. Имя должно быть представлено в форме "Источник.Поле". Если поле имеет значение null, оно преобразуется в значение по умолчанию (0, пустая строка, false и др).
`object GetColumnValueNullable(`` string complexName)`	Возвращает значение поля источника данных. В отличие от предыдущего метода, не выполняет преобразование null-значений.
`Parameter GetParameter(`` string complexName)`	Возвращает параметр отчета с указанным именем. Имя может быть составным при обращении к вложенному параметру: "MainParam.NestedParam".
`object GetParameterValue(`` string complexName)`	Возвращает значение параметра отчета с указанным именем.
`void SetParameterValue(`` string complexName, `` object value)`	Устанавливает значение параметра отчета с указанным именем.
`object GetVariableValue(`` string complexName)`	Возвращает значение системной переменной, например, "Date".
`object GetTotalValue(`` string name)`	Возвращает значение итога, определенного в окне "Данные", по его имени.
`DataSourceBase GetDataSource(`` string alias)`	Возвращает источник данных, определенный в отчете, по его имени.

Объект Engine представляет собой движок, управляющий построением отчета. Используя свойства и методы движка, можно управлять процессом размещения бэндов на странице. Вы можете использовать следующие свойства объекта Engine:

Свойство	Описание
`float CurX`	Текущее смещение координат по оси X. Этому свойству можно присваивать значение, чтобы смещать печатаемые объекты.
`float CurY`	Текущая позиция печати по оси Y. Этому свойству можно присваивать значение, чтобы смещать печатаемые объекты.
`int CurColumn`	Номер текущей колонки в многоколоночном отчете. Первая колонка имеет номер 0.

Свойство	Описание
`int CurPage`	Номер текущей печатаемой страницы. Это значение можно получить из системной переменной Page.
`float PageWidth`	Ширина страницы минус размер левого и правого полей.
`float PageHeight`	Высота страницы минус размер верхнего и нижнего полей.
`float PageFooterHeight`	Высота подвала страницы (и всех его дочерних бэндов).
`float ColumnFooterHeight`	Высота подвала колонки (и всех ее дочерних бэндов).
`float FreeSpace`	Размер свободного места на странице.
`bool FirstPass`	Возвращает true, если выполняется первый (или единственный) проход отчета. Количество проходов можно получить из свойства Report.DoublePass.
`bool FinalPass`	Возвращает true, если выполняется последний (или единственный) проход отчета.

На рисунке ниже представлено изображение страницы отчета и название свойств, которые возвращают то или иное измерение страницы.



Свойства Engine.PageWidth, Engine.PageHeight определяют размер области печати, которая почти всегда меньше физических размеров страницы. Размер области печати определяют поля страницы, которые задаются свойствами страницы отчета LeftMargin, TopMargin, RightMargin, BottomMargin.

Свойство Engine.FreeSpace определяет высоту свободного места на странице. Если на странице есть бэнд "Подвал страницы", его высота учитывается при вычислении FreeSpace. Следует учесть, что после вывода очередного бэнда свободное место на странице уменьшается, что учитывается при вычислении FreeSpace.

Как происходит формирование страниц готового отчета? Ядро FastReport выводит бэнды на страницу до тех пор, пока на ней остается свободное место, достаточное для вывода бэнда. Когда свободного места не остается, печатается бэнд "Подвал страницы" (если он есть) и формируется новая пустая страница. Как уже говорилось, после вывода очередного бэнда высота свободного места уменьшается. Кроме того, вывод очередного бэнда начинается с текущей позиции, которая определяется координатами по оси X и Y. Эта позиция возвращается в свойствах Engine.CurX, Engine.CurY. После печати очередного бэнда позиция CurY автоматически увеличивается на высоту напечатанного бэнда. После формирования новой страницы позиция CurY = 0. Позиция CurX изменяется при печати многоколоночных отчетов.

Свойства Engine.CurX, Engine.CurY доступны не только для чтения, но и для записи. Это значит, что можно смещать бэнды вручную, используя одно из подходящих событий. Пример использования этих свойств смотрите в разделе "Примеры".

> При работе со свойствами, которые возвращают размер, учтите, что в FastReport используются единицы измерения - экранные пикселы.

В объекте Engine определены следующие методы:

Метод	Описание
`void AddOutline(string text)`	Добавляет элемент в структуру отчета и смещает текущую позицию на добавленный элемент.
`void OutlineRoot()`	Смещает текущую позицию на корень структуры.
`void OutlineUp()`	Смещает текущую позицию на уровень выше.
`void AddBookmark(string name)`	Добавляет закладку.
`int GetBookmarkPage(string name)`	Возвращает номер страницы, на которой расположена закладка с указанным именем.
`void StartNewPage()`	Добавляет в отчет разрыв страницы. Если отчет многоколоночный, добавляется новая колонка.

Используя методы AddOutline, OutlineRoot, OutlineUp, можно формировать структуру отчета программным способом. Обычно это делается автоматически с помощью свойства OutlineExpression, которое имеется у каждого бэнда и у страницы отчета.

Метод AddOutline добавляет к текущему узлу структуры дочерний узел и делает его текущим. С элементом ассоциируется текущая страница отчета и текущая позиция на странице. Таким образом, если несколько раз подряд вызвать AddOutline, то получится "лесенка" типа

```
Item1
    Item2
        Item3
```

Для управления текущим элементом служат методы OutlineUp и OutlineRoot. Первый метод перемещает указатель на элемент, расположенный уровнем выше. Так, скрипт

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineUp();
Engine.AddOutline("Item4");
```

построит структуру вида

```
Item1
    Item2
        Item3
        Item4
```

Метод OutlineRoot передвигает текущий элемент в корень структуры. Например, скрипт:

```
Engine.AddOutline("Item1");
Engine.AddOutline("Item2");
Engine.AddOutline("Item3");
Engine.OutlineRoot();
Engine.AddOutline("Item4");
```

построит структуру следующего вида:

```
Item1
    Item2
        Item3
Item4
```

Для работы с закладками используются методы AddBookmark и GetBookmarkPage объекта Engine. Обычно закладки добавляются автоматически при использовании свойства Bookmark, которое имеется у всех объектов отчета.

Используя метод AddBookmark, можно добавлять закладку программно. Этот метод создаст закладку на текущей странице, в текущей позиции печати.

Метод GetBookmarkPage возвращает номер страницы, на которой расположена закладка. Этот метод часто применяется при создании оглавлений, для отображении номеров страниц. В этом случае отчет должен быть двухпроходным.

Обращение к источникам данных

В отличие от выражений FastReport (они рассмотрены в главе "Выражения"), в скрипте нельзя использовать квадратные скобки для обращения к данным отчета. Вместо этого используется метод GetColumnValue объекта Report, возвращающий значение поля:

```
string productName = (string)Report.GetColumnValue("Products.Name");
```

Как видно, надо указать имя источника данных и его поля через точку. Имя источника может быть составным в случае, если мы обращаемся к источнику данных, используя связь (relation). Например, так можно обратиться к полю связанного источника данных:

```
string categoryName = (string)Report.GetColumnValue("Products.Categories.CategoryName");
```

Для облегчения работы используйте окно "Данные". Из него можно перетаскивать элементы данных в скрипт, при этом FastReport автоматически создает код для обращения к элементу.

Для обращения к самому источнику данных используйте метод GetDataSource объекта Report:

```
DataSourceBase ds = Report.GetDataSource("Products");
```

Справку по свойствам и методам класса DataSourceBase можно получить в справочной системе FastReport.Net Class Reference. Как правило, этот объект используется в скрипте следующим образом:

```
// получаем ссылку на источник данных
DataSourceBase ds = Report.GetDataSource("Products");
// инициализируем его
ds.Init();
// перебираем все записи в источнике
while (ds.HasMoreRows)
{
  // получаем значение поля для текущей записи источника
  string productName = (string)Report.GetColumnValue("Products.Name");
  // выполняем с ним какие-то действия...
  // ...
  // переходим на следующую запись
  ds.Next();
}
```

Обращение к системным переменным

Для обращения к системной переменной используйте метод GetVariableValue объекта Report:

```
DateTime date = (DateTime)Report.GetVariableValue("Date");
```

Список системных переменных можно увидеть в окне "Данные". Из него можно перетаскивать переменные в скрипт, при этом FastReport автоматически создает код для обращения к переменной.

Обращение к итоговым значениям

Для обращения к итоговому значению используйте метод GetTotalValue объекта Report:

```
float sales = Report.GetTotalValue("TotalSales");
```

Список итогов можно увидеть в окне "Данные". Из него можно перетаскивать итоги в скрипт, при этом FastReport автоматически создает код для обращения к итогу.

Итоговое значение имеет тип `FastReport.Variant`. Оно может быть напрямую использовано в любых выражениях, потому что тип `FastReport.Variant` автоматически приводится к любому типу. Например:

```
float tax = Report.GetTotalValue("TotalSales") * 0.2f;
```

Обращаться к итоговому значению можно в тот момент, когда оно вычислено. Обычно итог "готов к употреблению" в момент печати бэнда, на котором он располагается в отчете.

Обращение к параметрам отчета

Для обращения к параметру отчета используйте метод GetParameterValue объекта Report:

```
int myParam = (int)Report.GetParameterValue("MyParameter");
```

Параметры могут быть вложенными. В этом случае укажите имя родительского параметра и через точку имя дочернего параметра:

```
Report.GetParameterValue("ParentParameter.ChildParameter")
```

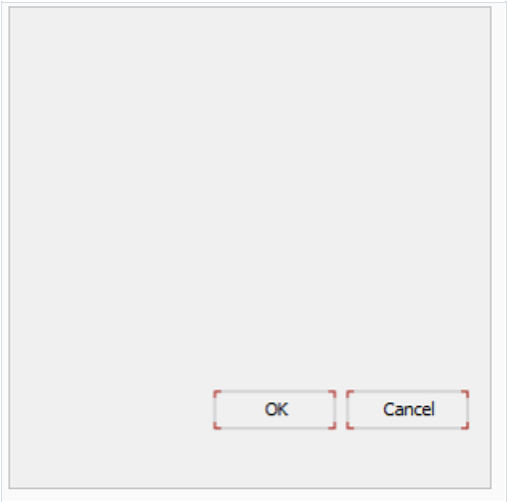
Параметры имеют определенный тип данных. Он задается в свойстве DataType параметра. Вы должны учитывать это при обращении к параметру. Список параметров можно увидеть в окне "Данные". Из него можно перетаскивать параметры в скрипт, при этом FastReport автоматически создает код для обращения к параметру.

Для изменения значения параметра используйте метод SetParameterValue объекта Report:

```
Report.SetParameterValue("MyParameter", 10);
```

Диалоговые формы

Кроме обычных страниц, отчет может содержать диалоги. Диалог - это окно с элементами управления, которое показывается в момент запуска отчета. В диалоге можно ввести какую-либо информацию, необходимую для построения отчета. Также диалог удобно использовать для фильтрации данных, выводимых в отчете.



Чтобы добавить диалоговую форм в отчет, нужно открыть вкладку «Отчет» и нажать на пиктограмму 

. Для удаления диалога, нужно перейти на закладку с диалоговой формой и на вкладке «Отчет» нажать на пиктограмму 

.

Отчет, который содержит один или несколько диалогов, работает так:

- при запуске отчета показывается первый диалог;
- если диалог закрывается с помощью кнопки "OK", показывается следующий диалог;
- если диалог закрывается с помощью кнопки "Отмена" или крестиком на заголовке окна, работа отчета завершается;
- после того как показаны все диалоги, выполняется построение отчета.

Элементы управления

На диалоговой форме можно использовать следующие элементы управления:

Иконка	Название	Описание
	ButtonControl	Представляет собой кнопку.
	CheckBoxControl	Представляет собой флажок, имеющий два состояния (включен/выключен).
	CheckedListBoxControl	Представляет собой список строк с флажками.
	ComboBoxControl	Представляет собой выпадающий список.
	DateTimePickerControl	Позволяет ввести дату или время.
	LabelControl	Представляет собой поясняющую надпись.
	ListBoxControl	Представляет собой список строк.
	MonthCalendarControl	Представляет собой календарь.
	RadioButtonControl	Представляет собой зависимый переключатель, имеющий два состояния (включен/выключен).
	TextBoxControl	Представляет собой поле для редактирования однострочного или многострочного текста.

Все элементы управления являются полными аналогами стандартных элементов управления, доступных в .Net Framework. Названия элементов имеют приставку Control, чтобы избежать совпадения имен. Так, элементу управления ButtonControl соответствует элемент Button из .Net Framework.

Обратиться к элементу из кода можно, используя его имя:

```
TextBoxControl1.Text = "my text";
```

По сути, элемент управления FastReport является оберткой над стандартным элементом управления .Net Framework. Он реализует многие, но не все, свойства стандартного элемента. Если реализованных свойств вам не достаточно, вы можете обратиться к стандартному элементу следующим образом:

- используя свойство Control, которое возвращает тип System.Windows.Forms.Control:

```
(TextBox1.Control as TextBox).ShortcutsEnabled = false;
```

- используя свойство, имя которого совпадает с именем элемента без приставки "Control". Например, для элемента TextBoxControl свойство TextBox возвращает стандартный элемент типа System.Windows.Forms.TextBox:

```
TextBox1.TextBox.ShortcutsEnabled = false;
```

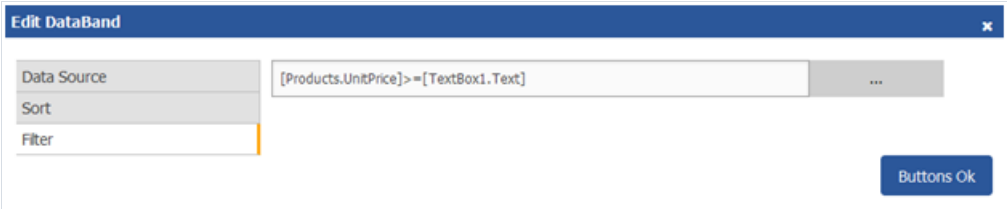
Справку по свойствам и методам элементов управления можно получить в MSDN.

Фильтрация данных

Один из способов использования диалогов - это фильтрация данных, выводимых в отчете. Например, имеется отчет, печатающий список всех сотрудников. Используя диалог, можно дать пользователю возможность выбора одного или нескольких из них, затем при построении отчета отфильтровать данные, чтобы показать выбранных сотрудников.

> Для использования фильтрации необходимо, чтобы исходный отчет печатал все данные. Само название "фильтрация" предполагает, что ненужные данные будут отброшены в процессе построения отчета.

Самый простой способ организовать фильтрацию данных - использовать свойство Filter бэнда "Данные". В редакторе бэнда можно указать условие фильтрации, например:



Используя диалог, можно запросить у пользователя значение и использовать его в выражении фильтра.

Этот способ можно использовать, если нужно запросить одно простое значение. Если же стоит задача показать пользователю список значений и запросить одно или несколько из них, реализовать это становится довольно тяжело. Казалось бы, простая задача - показать пользователю список сотрудников в элементе управления ListBoxControl и выбрать одно или несколько значений. Для реализации этого надо использовать скрипт, который выполнит следующее:

- получит источник данных по его имени;
- инициализирует данные;
- заполнит список ListBoxControl имеющимися в источнике значениями;
- после выбора сотрудников сформирует условие для фильтрации.

FastReport позволяет выполнить все эти действия автоматически, без написания какого-либо кода. Для этого используется автоматическая фильтрация, которую мы сейчас рассмотрим.

Автоматическая фильтрация

Элемент управления подключается к полю данных с помощью свойства DataColumn. Если элемент может отображать список значений (например, ListBoxControl), он заполняется всеми значениями из указанного поля данных. Это происходит автоматически при запуске диалога. Далее пользователь работает с диалогом, выбирает одно или несколько значений в элементе и закрывает диалог. В этот момент к источнику данных, который указан в свойстве DataColumn, применяется фильтр, содержащий выбранные значения.

> Преимущество данного способа заключается в том, что его можно использовать в любом отчете, не написав для этого ни строки кода.

Автоматическая фильтрация поддерживается следующими элементами управления:

Иконка	Название
	CheckBoxControl
	CheckedListBoxControl
	ComboBoxControl
	DateTimePickerControl
	ListBoxControl
	MonthCalendarControl
	RadioButtonControl
	TextBoxControl

Операция фильтра

По умолчанию FastReport фильтрует строки данных, которые содержат значения, равные значению элемента управления. Это настраивается с помощью свойства "FilterOperation" элемента управления. В этом свойстве указывается операция которая будет использована при фильтрации данных. Вы можете использовать следующие операции:

Операция	Эквивалент	Действие
Equal	=	Выбрать значения, равные значению элемента управления.
NotEqual	<>	Выбрать значения, не равные значению элемента.
LessThan	<	Выбрать значения, меньшие чем значение элемента.
LessThanOrEqual	<=	Выбрать значения, меньшие чем или равные значению элемента.
GreaterThan	>	Выбрать значения, бОльшие чем значение элемента.
GreaterThanOrEqual	>=	Выбрать значения, бОльшие чем или равные значению элемента.

Например, если свойство FilterOperation элемента управления установлено в LessThanOrEqual и вы ввели в элемент значение 5, то будут выбраны все строки данных, для которых значение фильтруемого поля меньше или равно 5.

Для данных строкового типа можно использовать дополнительные операции, которые проверяют вхождение одной строки в другую:

Операция	Действие
Contains	Выбрать строки, содержащие значение из элемента управления.
NotContains	Выбрать строки, не содержащие значение из элемента.
StartsWith	Выбрать строки, начинающиеся со значения из элемента.
NotStartsWith	Выбрать строки, не начинающиеся со значения из элемента.
EndsWith	Выбрать строки, заканчивающиеся значением из элемента.
NotEndsWith	Выбрать строки, не заканчивающиеся значением из элемента.

Например, если свойство FilterOperation элемента управления установлено в StartsWith и вы ввели в элемент значение A, то будут выбраны все строки данных, для которых значение фильтруемого поля начинается на A.

Фильтрация по диапазону значений

Этот способ фильтрации удобно использовать при работе со значениями, имеющими количественную характеристику, например, стоимость. Таким образом можно отфильтровать товар, имеющий стоимость ниже или выше заданной. Для того чтобы указать, как следует трактовать введенное в элемент значение, используется свойство FilterOperation, рассмотренное выше.

Используя два элемента управления, которые подключены к одному и тому же полю данных и имеют разные настройки свойства FilterOperation, можно указать начало и конец диапазона данных. Для первого элемента надо указать свойство FilterOperation = GreaterThanOrEqual, для второго - LessThanOrEqual.

Фильтрация по связанному полю

Как нам известно, между двумя источниками данных можно установить связь. С помощью связи можно фильтровать данные в источнике, используя для этого поле из другого, связанного источника.

Допустим, вы положили на диалог элемент управления ListBoxControl и указали в свойстве DataColumn следующее поле данных:

```
Products.Categories.CategoryName
```

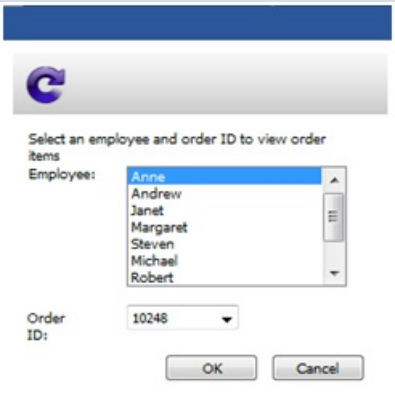
Как будет работать фильтрация?

- при заполнении элемента управления будет использовано поле CategoryName источника данных Categories;
- при фильтрации данных будут отфильтрованы те строки источника данных Products, для которых верно условие:

```
поле [Products.Categories.CategoryName] содержит одно из выбранных пользователем значений.
```

Фильтрация с каскадными списками

Каскадный список - это список, набор значений в котором определяется выбором, сделанным пользователем в другом списке. Допустим, на форме есть два списка - список категорий и список продуктов. При выборе категории из первого списка, во втором списке будут отображены только те продукты, которые входят в выбранную категорию.



Для создания каскадного списка нужно два источника данных, связанных отношением типа "главный-подчиненный". Первый список (главный) прикрепляется к полю из главного источника данных, второй список (подчиненный) - к полю подчиненного источника. Кроме того, надо настроить свойство DetailControl главного списка, указав в нем подчиненный список.

> Каскадную фильтрацию удобно применять к отчету типа master-detail. Если отчет другого типа (например, с одним бэндом "Данные"), для корректной работы фильтрации нужно будет использовать скрипт.

Управление фильтрацией из кода

Хотя возможностей автоматической фильтрации достаточно для большинства случаев, у вас есть возможность управлять фильтрацией вручную. Для этого используются следующие свойства и методы.

Свойство AutoFill управляет заполнением элемента. Оно используется элементами, которые могут показывать список данных, например, ListBoxControl. Перед тем как показать диалог, FastReport заполняет данными такие элементы. По умолчанию свойство равно true. Если его отключить, элемент заполняться не будет, и вы должны сделать это сами, вызвав метод FillData:

```
ListBox1.FillData();
```

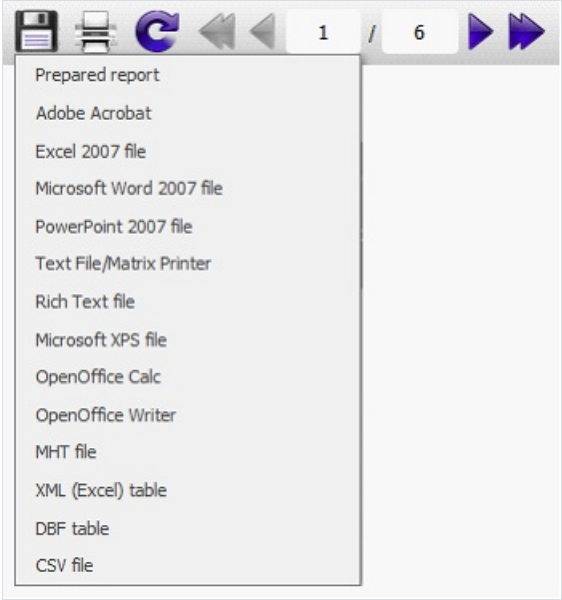
Свойство AutoFilter управляет фильтрацией данных. Оно используется всеми элементами управления. После того как диалог закрыт кнопкой "OK", FastReport выполняет фильтрацию данных автоматически. По умолчанию свойство равно true. Если его отключить, фильтрация выполняться не будет, и вы должны сделать это сами, вызвав метод FilterData:

```
ListBox1.FilterData();
```

Экспорт отчета

Построенный отчет можно сохранить в одном из форматов: файл подготовленного отчета (fpx), Adobe Acrobat (PDF), Microsoft Word 2007, Excel 2007, PowerPoint 2007, TXT, RTF, Microsoft XPS, OpenOffice Calc, OpenOffice Writer, MHT, HTML, Excel XML, DBF, CSV.

Чтобы выбрать формат экспорта, запустите отчет в режиме предварительного просмотра, наведите курсор мыши на значок дискеты, и выберите один из форматов в выпадающем списке:



При экспорте будет предложено стандартное окно сохранения файла.

Также в режиме предварительного просмотра можно распечатать отчет. Для этого нажмите на значок принтера на верхней панели инструментов. При этом откроется стандартное окно печати.

Сохранение в формате FPX

Формат .FPX - это "родной" формат FastReport. Преимущества этого формата заключаются в следующем:

- сохранение отчета без потери качества. Открыв сохраненный ранее файл, вы можете выполнять над ним все операции, такие как печать, экспорт, редактирование;
- компактный формат хранения - XML, сжатый с помощью ZIP;
- при необходимости файл отчета можно распаковать любым архиватором, поддерживающим ZIP-формат и поправить вручную в любом текстовом редакторе.

Недостаток формата заключается только в том, что для его просмотра необходимо наличие FastReport.

Экспорт в AdobeAcrobat (PDF)

PDF (Portable Document Format) – платформо-независимый формат электронных документов, созданный фирмой Adobe Systems. Для просмотра используется бесплатный пакет Acrobat Reader. Данный формат достаточно гибкий - позволяет внедрять необходимые шрифты, векторные и растровые изображения, очень хорошо подходит для передачи и хранения документов, предназначенных для просмотра и последующей печати.

Метод экспорта – послойный.

Экспорт в Excel 2007

Excel 2007 – приложение для работы с электронными таблицами, включенное в систему Microsoft Office 2007.

Метод экспорта – табличный.

Экспорт в Microsoft Word 2007 (DOCX)

Microsoft Word 2007 – текстовый редактор, включенный в пакет программ Microsoft Office 2007.

Метод экспорта – табличный.

Экспорт в PowerPoint 2007

PowerPoint 2007 – приложение для работы с презентациями, включенное в систему Microsoft Office 2007.

Метод экспорта – послойный.

Экспорт в TXT

Обычный текстовый файл – содержит информацию из отчета, максимально оптимизированную и преобразованную в связи со спецификой данного формата. Полученный документ может быть сохранен в файл или распечатан на матричном принтере.

Метод экспорта – табличный.

Экспорт в RTF

RTF (Rich Text Format) был разработан фирмой Microsoft как стандартный формат для обмена текстовыми документами. На данный момент RTF-документы совместимы с большинством современных текстовых редакторов и операционных систем.

Метод экспорта – табличный.

Экспорт в Microsoft XPS

Формат XPS - графический формат фиксированной разметки данных, основанный на XML. Файл XPS подобен файлу формата PDF, но вместо PostScript для описания разметки и метаданных используется XML.

Метод экспорта – послойный.

Экспорт в OpenOffice Calc

Open Document Format - открытый формат файлов документов для хранения и обмена редактируемыми офисными документами, в том числе текстовыми документами (такими как заметки, отчёты и книги), электронными таблицами, рисунками, базами данных, презентациями. Этот стандарт был разработан индустриальным сообществом OASIS и основан на XML-формате, изначально созданном OpenOffice.org. 1 мая 2006 года принят как международный стандарт ISO/IEC 26300.

FastReport поддерживает экспорт в таблицу (расширение .ods) Open Documet. Эти файлы могут быть открыты с помощью бесплатного офисного пакета OpenOffice.

Метод экспорта – табличный.

Экспорт в OpenOffice Writer

OpenOffice Writer – текстовый процессор с открытым кодом состава пакета OpenOffice. Подобен редактору Microsoft Office Word. Имеет собственный формат файлов .odt.

Метод экспорта – табличный.

Экспорт в MHT (веб-архив)

Формат MHT (веб-архив) основан на MIME кодировании и позволяет хранить в одном файле текстовый и графический контент страницы. Сохранение страниц в MHT поддерживается Internet Explorer.

Метод экспорта – табличный.

Экспорт в Excel (XML)

Excel – приложение для работы с электронными таблицами, включенное в систему Microsoft Office.

Метод экспорта – табличный.

Экспорт в DBF

Формат DBF предназначен для хранения и передачи данных и является стандартным способом хранения в некоторых СУБД. Также, поддерживается многими табличными редакторами, например Microsoft Excel.

Метод экспорта – табличный.

Экспорт в CSV

CSV-файл содержит значения, отформатированные в виде таблицы и упорядоченные таким образом, что каждое значение в столбце отделено от значения в следующем столбце разделителем, а каждый новый ряд начинается с новой строки. Данный формат может быть импортирован в различные табличные редакторы.

Метод экспорта – табличный.