Руководство программиста FastReport (.NET, Mono, WPF)

Версия 2024.1

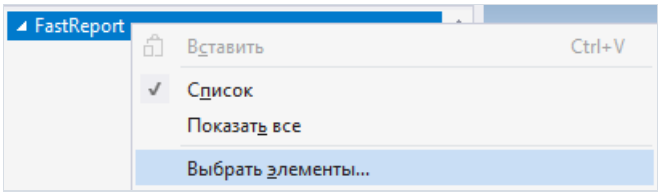
Общая информация

Установка в Toolbox

При работе с пакетами FastReport.Net* компоненты Toolbox включены в пакет и должны подгружаться Visual Studio автоматически. Также возможна ручная установка Toolbox.

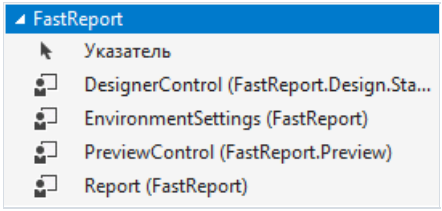
Для ручной установки компонентов FastReport .NET в Visual Studio Toolbox сделайте следующее:

- удалите категорию "FastReport .NET" из Toolbox, если она присутствует;
- создайте новую категорию (для этого щелкните правой кнопкой мыши на Toolbox и выберите пункт меню "Add Tab"), либо выберите уже существующую категорию, в которую вы хотите поместить компоненты FastReport;
- щелкните правой кнопкой мыши на выбранной категории и выберите пункт меню "Choose Items...":



- в открывшемся окне нажмите кнопку "Browse..." и выберите файлы FastReport.dll, FastReport.Web.dll (они находятся в папке "C:\Program Files\FastReports\FastReport.NET");
- закройте окно кнопкой OK.

После этого в выбранной вами категории появятся следующие компоненты FastReport:



- Report;
- PreviewControl;
- EnvironmentSettings;
- DesignerControl;
- WebReport (этот компонент будет доступен только при работе с проектом ASP.NET).

Устранение неполадок

В случае проблем в работе дизайнера (например, "съехали" панели инструментов) может понадобиться удаление файла конфигурации. Этот файл создается во время работы FastReport и находится в папке:

Windows XP:

```
C:\Documents and Settings\Имя_пользователя\Local Settings\Application Data\FastReport
```

Windows Vista и выше:

```
C:\Users\Имя_пользователя\AppData\Local\FastReport
```

Имя файла зависит от продукта:

- FastReport.config для FastReport .NET;
- FastReport.Mono.config для FastReport.Mono;
- FastReport.WPF.config для FastReport WPF.

В файле конфигурации хранится следующая информация:

- размеры и расположение окон интерфейса;
- настройки панелей инструментов;
- параметры последних использованных подключений к БД;
- настройки email (при использовании функции "Отправка email" в окне просмотра отчета).

Распространение

Вы можете распространять следующие файлы вместе со своим приложением:

- FastReport.dll (FastReport.Mono.dll, FastReport.WPF.dll) - основная библиотека FastReport;
- FastReport.Web.dll - библиотека для работы в ASP.Net, содержит компонент WebReport;
- FastReport.Bars.dll - библиотека для организации плавающих окон, панелей инструментов и меню (относится к FastReport .NET);
- FastReport.Editor.dll - редактор кода с подсветкой синтаксиса. Эта библиотека не нужна, если ваше приложение не использует дизайнер отчетов (относится к FastReport .NET);
- FastReport.WPF.RoslynPad.dll - редактор кода с подсветкой синтаксиса для .Net 6.0+ (относится к FastReport WPF);
- FastReport.Forms.WPF.dll - библиотека для поддержки форм (относится к FastReport WPF);
- FastReport.SkiaDrawing.dll - библиотека для поддержки Skia;
- FastReport.Compat.dll (FastReport.Compat.Skia.dll) - библиотека для кроссплатформенной совместимости;
- FastReport.DataVisualization.dll (FastReport.DataVisualization.Skia.dll) - библиотека отрисовки диаграмм;
- FastReport.xml (FastReport.Mono.xml, FastReport.WPF.xml) - комментарии к классам, свойствам и методам FastReport. Этот файл используется в редакторе кода, а также в панелях подсказки (когда вы выбираете функцию в окне "Данные" или любое свойство в окне "Свойства"). Этот файл распространять не обязательно.

Кроме того, вам необходимо распространять файлы отчетов (если отчеты хранятся в файлах, а не в ресурсах приложения).

FastReport NuGet пакеты:

- **FastReport.Core (демо на nuget.org)** - пакет с основной логикой работы программы (получение необходимых данных, рендер отчётов, экспорты и т.д). Часть функциональности из FastReport.NET отсутствует в связи с кроссплатформенностью пакета.
- **FastReport.Net (демо на nuget.org)** - пакет с библиотекой FastReport.dll для .NET Framework 4.x, который входит в состав 'Pro' и 'Demo' изданий - для .NET Core 3.1, .NET 5 и .NET 6 исключительно под Windows (так называемый FastReport.CoreWin). Этот пакет доступен в разных редакциях:
  - FastReport.NET.Demo - Данный пакет доступен на nuget.org и в нашем официальном установщике Trial версии и позволяет оценить возможности продукта для разных целевых платформ на ОС Windows. В этом пакете предусмотрены ограничения, присущие для Demo редакции (Watermark и др.).
  - FastReport.NET - Данный пакет доступен в нашем официальном установщике для обладателей лицензионных редакций FastReport .NET WinForms и Web. В него входит FastReport .NET для .NET Framework 4.0 и выше.
  - FastReport.NET.Pro - Данный пакет доступен в нашем официальном установщике для обладателей лицензионных редакций FastReport .NET Professional и Enterprise. В него входит FastReport .NET для .NET Framework 4.0 и выше, FastReport.CoreWin для .NET Core 3.1 и для .NET 5.0 и выше.
- **FastReport.WPF (демо на nuget.org)** - пакет с библиотекой FastReport WPF;
- **FastReport.Web (демо на nuget.org)** - пакет для интеграции FastReport в сценарии работы с веб-приложениями (рендер отчёта в браузере, экспорт и печать из браузера, работа с Online Designer) для ASP.NET Core. Включает в себя компоненты для Blazor Server и используется только с FastReport.Core.
- **FastReport.Core3.Web (демо на nuget.org)** - тот же принцип, что и FastReport.Web, но совместим с FastReport.CoreWin, который идёт в составе пакета FastReport.Net.Demo / FastReport.Net.Pro.
- **FastReport.Localization (nuget.org)** - пакет с набором локализаций FastReport. Добавьте его в свой проект, если вам нужна, например, немецкая или русская локализация.

- **FastReport.Compat and FastReport.DataVisualization** - пакеты с базовой логикой (компиляция отчета, поддержка MSChart и т.п.). Включать их в свой проект не нужно, они являются пакетами-зависимостями.
- **FastReport.Data.*** - пакеты с плагинами-коннекторами для работы FastReport с различными базами данных, коннекторы которых не включены в исходную библиотеку. Данные пакеты "общие" для разных редакций FastReport и подойдут как к FastReport .NET, так и к FastReport.Core и FastReport.CoreWin. Ограничения: необходима версия FastReport 2021.4.0+ и NuGet Client 3.4.4+.
    - FastReport.Data.ClickHouse
    - FastReport.Data.Couchbase
    - FastReport.Data.Firebird
    - FastReport.Data.Json
    - FastReport.Data.MongoDB
    - FastReport.Data.MsSql
    - FastReport.Data.MySql
    - FastReport.Data.OracleODPCore
    - FastReport.Data.Postgres
    - FastReport.Data.RavenDB
    - FastReport.Data.SQLite

Компиляция исходного кода

В комплект редакций Professional и Enterprise включены исходные коды библиотек FastReport. Вы можете включить их в состав своего приложения. Покажем, как сделать это на примере проекта FastReport .NET:

- откройте свой проект в Visual Studio;
- откройте окно Solution Explorer и щелкните правой кнопкой мыши на элементе "Solution";
- выберите пункт меню "Add/Existing Project...";
- добавьте файл проекта "FastReport.csproj" (он находится в папке "C:\Program Files\FastReports\FastReport.NET\Source\FastReport");

Обновите ссылки на другие сборки FastReport .NET. Для этого:

- раскройте элемент "FastReport\References" в окне Solution Explorer;
- удалите ссылки на "FastReport.Bars", "FastReport.Editor";
- щелкните правой кнопкой мыши на элементе "References";
- выберите пункт меню "Add Reference...";
- добавьте ссылки на сборки "FastReport.Bars.dll" и "FastReport.Editor.dll". Эти файлы находятся в папке "C:\Program Files\FastReports\FastReport.NET".

Для сборки FastReport WPF из исходного кода откройте файл "C:\Program Files\FastReports\FastReport WPF\Sources\FastReport.WPF\FastReport.WPF.sln". В состав решения входят все проекты FastReport WPF, а также демо-приложения (Samples).

Работа в Windows.Forms и WPF

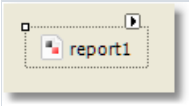
Использование компонента Report в Windows.Forms

Рассмотрим типичный сценарий использования компонента Report в среде Visual Studio. При этом данные берутся из типизированного источника.

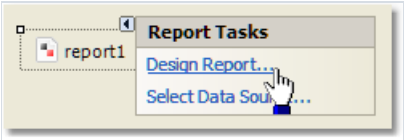
- создайте проект Windows Forms Application;
- добавьте в него источник данных (в меню "Data|Add New Data Source...");
- переключитесь на дизайнер формы;
- добавьте на форму компонент DataSet из категории "Data" и подключите его к типизированному источнику данных, который вы создали ранее.

Чтобы создать отчет, выполните следующие шаги:

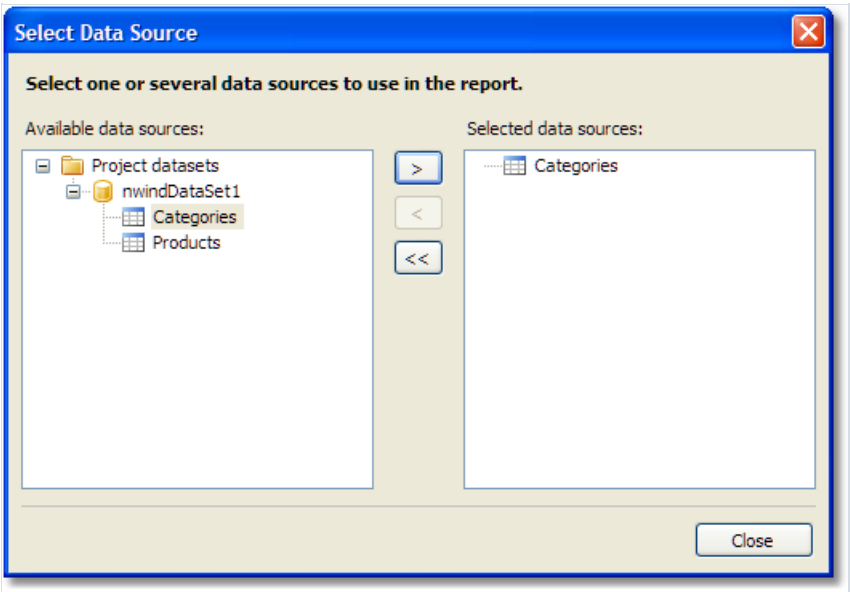
- положите на форму компонент Report:



- сделайте щелчок правой кнопкой мыши на компоненте, либо нажмите кнопку "smart tag", и выберите пункт "Design Report...":



- вам будет предложено выбрать источник данных для отчета. Выберите нужную таблицу (одну или несколько):



- создайте отчет. Подробнее об этом можно прочитать в "Руководстве пользователя";
- закройте дизайнер отчета;
- добавьте кнопку (Button) на форму и сделайте двойной щелчок на ней;
- в обработчике события напишите:

```
report1.Show();
```

- сохраните проект и запустите его. При нажатии на кнопку будет построен отчет.

Работа с отчетом в коде

Для работы с отчетом в коде, вам нужно выполнить следующие шаги:

- создать экземпляр отчета;
- загрузить в него файл отчета;
- зарегистрировать данные, определенные в приложении;
- передать в отчет значения параметров (если необходимо);
- запустить отчет на выполнение.

Следующий пример демонстрирует это:

```
using (Report report = new Report())
{
  report.Load("report1.frx");
  report.RegisterData(dataSet1, "NorthWind");
  report.Show();
}
```

Далее мы рассмотрим подробнее каждый из шагов.

Хранение и загрузка отчета

Вы можете хранить отчет одним из следующих способов:

Способ	Описание
в ресурсах приложения (FastReport .NET/Mono)	Типичный сценарий работы с отчетом, который мы рассмотрели выше, использует именно этот способ. За это отвечает свойство StoreInResources объекта Report, которое по умолчанию равно true. У данного способа есть следующие плюсы и минусы: + отчет встроен в приложение, не нужно распространять дополнительные файлы; - при необходимости изменить отчет, нужно перекомпилировать приложение. Загрузка отчета осуществляется автоматически с помощью кода, который FastReport .NET добавляет в метод InitializeComponent вашей формы.
в файле .FRX	Это наиболее подходящий способ, если вы хотите дать пользователям возможность редактировать файлы отчета. В этом случае установите свойство StoreInResources в false. Для загрузки отчета из файла используйте метод Load объекта Report: `report1.Load("filename.frx");`
в базе данных	Отчет можно хранить в поле БД в виде строки, либо в двоичном потоке. Для загрузки отчета из строки используйте метод LoadFromString объекта Report. Для загрузки из потока используйте перегруженный метод Load: `report1.Load(stream);` Для поддержки этого способа в дизайнере отчетов необходимо переопределить реакцию на действия "Открыть файл" и "Сохранить файл". Как это сделать, описано ниже.

в виде класса C#/VB.NET	Для этого в дизайнере отчета выберите пункт "Файл/Сохранить как...", и выберите тип файла - "Файл C#" или "Файл VB.Net" (это зависит от того, какой язык выбран в самом отчете - см. "Руководство пользователя"). Полученный файл можно добавить в ваш проект. У данного способа есть следующие плюсы и минусы:
	+ работа с отчетом, как с обычным классом;
	+ доступны все возможности Visual Studio, в том числе отладка;
	+ это единственный способ работы с отчетом, который совместим с режимом Medium Trust в ASP.NET;
	- вы не можете редактировать такой отчет. Для этого нужно иметь оригинальный файл отчета в формате .FRX;
	- при необходимости изменить отчет, нужно перекомпилировать приложение.
	Для работы с отчетом создайте экземпляр класса отчета, например:
	`SimpleListReport report = new SimpleListReport();`
	`report.Show();`

Регистрация данных

Если ваш отчет использует данные из приложения (например, типизированный источник данных или бизнес-объект), то их необходимо зарегистрировать в отчете. Это делается с помощью метода RegisterData объекта Report.

> При типичном сценарии работы с отчетом, рассмотренном здесь, вам не нужно регистрировать данные вручную. FastReport автоматически добавляет вызов метода RegisterData в код метода InitializeComponent вашей формы.

Метод RegisterData нужно вызывать после того, как отчет загружен:

```
report1 = new Report();
report1.Load("report.frx");
report1.RegisterData(dataSet1, "NorthWind");
```

Метод RegisterData перегружен и позволяет регистрировать следующие типы данных:

Метод	Описание
`void RegisterData(DataSet data)`	Регистрация источника данных. Этот метод регистрирует все таблицы, представления и связи, которые имеются в источнике. Вызов этого метода эквивалентен вызову RegisterData(data, "Data"). Внимание: если вы регистрируете несколько источников, используйте версию метода `RegisterData(DataSet data, string name)`.
`void RegisterData(DataSet data, string name)`	Регистрация источника данных. Этот метод регистрирует все таблицы, представления и связи, которые имеются в источнике. В параметре name можно указать любое имя; важно, чтобы оно не менялось и было уникальным (в случае, если регистрируется несколько источников). Имя name нигде не отображается, однако оно используется для привязки объектов отчета к набору данных. Например, мы регистрируем набор данных dataSet1, содержащий две таблицы - Customers и Orders: `report.RegisterData(dataSet1, "MyDataSet");` В файл отчета (.frx) будет добавлено два источника данных следующего вида: `<TableDataSource Name="Customers" ReferenceName="MyDataSet.Customers">` `...` `</TableDataSource>` `<TableDataSource Name="Orders" ReferenceName="MyDataSet.Orders">` `...` `</TableDataSource>` Как видно, они привязаны к исходному набору данных с помощью свойства ReferenceName. Теперь, если загрузить этот отчет и вызвать метод `report.RegisterData(dataSet1, "MyDataSet");` будет выполнена привязка объектов TableDataSource в отчете к соответствующим таблицам набора данных dataSet1. Если привязку осуществить не удалось (например, указано другое имя name), при запуске отчета будет выдано окно с ошибкой "Источник данных не инициализирован".
`void RegisterData(DataTable data, string name)`	Регистрация таблицы.
`void RegisterData(DataView data, string name)`	Регистрация представления (view).

Метод	Описание
`void RegisterDataAsp(IDataSource data, string name)`	Регистрация источника данных при работе в ASP.NET.
`void RegisterData(DataRelation data, string name)`	Регистрация связи.
`void RegisterData(IEnumerable data, string name)`	Регистрация бизнес-объекта. Этот вызов эквивалентен вызову `RegisterData(data, name, 1)`. Используя меню "Данные|Выбрать данные для отчета...", можно раскрыть структуру бизнес-объекта до нужной глубины вложенности.
`void RegisterData(IEnumerable data, string name, int maxNestingLevel)`	Регистрация дерева бизнес-объектов с начальным уровнем вложенности `maxNestingLevel`. Используя меню "Данные|Выбрать данные для отчета...", можно раскрыть структуру бизнес-объекта до нужной глубины вложенности.

Передача параметра в отчет

В отчете могут быть определены параметры. Подробнее об этом можно почитать в "Руководстве пользователя". Для передачи значения параметра используется метод SetParameterValue объекта Report:

```
report1.Load("report.frx");
report1.SetParameterValue("MyParam", 10);
report1.Show();
```

Метод определен следующим образом:

```
publicvoid SetParameterValue(string complexName, object value)
```

В параметре complexName необходимо указать имя параметра. Для обращения к вложенному параметру, используйте его полное имя, например:

```
"ParentParameter.ChildParameter"
```

Запуск отчета

Для запуска отчета используйте один из следующих методов объекта Report:

Метод	Описание

`void Show()`	Запускает отчет на построение и показывает его в окне предварительного просмотра. Этот метод равнозначен следующему:
	`if (Prepare())`
	`ShowPrepared();`

`bool Prepare()`	Запускает отчет на выполнение. Если отчет построен успешно, возвращает true. После этого метода нужно вызвать один из методов: ShowPrepared, PrintPrepared, SavePrepared, Export:
	`if (Prepare())`
	`ShowPrepared();`

`bool Prepare(bool append)`	Запускает отчет на выполнение. Если параметр append равен true, готовый отчет дописывается к уже имеющемуся. Таким образом, можно построить несколько отчетов и показать их в окне просмотра как один:
	`report1.Load("report1.frx");`
	`report1.Prepare();`
	`report1.Load("report2.frx");`
	`report1.Prepare(true);`
	`report.ShowPrepared();`

`void ShowPrepared()`	Показывает готовый отчет в окне предварительного просмотра. Отчет должен быть приготовлен с помощью метода Prepare, либо загружен из файла .FPX с помощью метода LoadPrepared:
	`if (Prepare())`
	`ShowPrepared();`

`void ShowPrepared(bool modal)`	Показывает готовый отчет в окне предварительного просмотра. Параметр modal определяет, надо ли показывать окно модально.
`void ShowPrepared(bool modal, Form owner)`	То же, что и предыдущий метод. Параметр owner определяет окно - владельца окна просмотра.
`void ShowPrepared(Form mdiParent)`	То же, что и предыдущий метод. Параметр mdiParent определяет главное окно MDI.

Запуск дизайнера

Во всех версиях FastReport есть возможность вызывать дизайнер отчета из вашей программы. Для этого используйте метод Design объекта Report:

```
report1 = new Report();
report1.Load("report1.frx");
report1.Design();
```

Метод Design перегружен:

Метод	Описание
`bool Design()`	Вызов дизайнера.
`bool Design(bool modal)`	То же. Параметр modal определяет, надо ли показывать окно дизайнера модально.
`bool Design(Form mdiParent)`	То же. Параметр mdiParent определяет главное окно MDI (относится к FastReport .NET/Mono).

Экспорт отчета

Готовый отчет может быть экспортирован в другие форматы. На настоящий момент поддерживаются следующие форматы:

- PDF
- HTML
- MHT
- RTF
- Excel XML (Excel 2003+)
- Excel 2007
- PowerPoint 2007
- CSV
- TXT
- OpenOffice Calc
- Изображения (bmp, png, jpeg, gif, tiff, metafile)

Экспорт отчета делается с помощью фильтра экспорта. Для этого нужно выполнить следующие шаги:

- построить отчет с помощью метода Prepare;
- создать экземпляр фильтра экспорта и настроить его свойства;
- вызвать метод Export фильтра, передав в него экземпляр отчета и имя файла.

Следующий пример показывает, как экспортировать отчет в формат HTML:

```
// готовим отчет
report1.Prepare();
// создаем экземпляр экспорта в HTML
FastReport.Export.Html.HTMLExport export = new FastReport.Export.Html.HTMLExport();
// показываем диалог с настройками экспорта и экспортируем отчет
if (export.ShowDialog())
  export.Export(report1, @"C:\result.html");
```

В примере настройка параметров экспорта осуществляется в диалоге.

Конфигурация среды FastReport

Используя глобальный класс FastReport.Utils.Config, вы можете поменять некоторые настройки среды FastReport.

Настройки, относящиеся к отчету, доступны в свойстве Config.ReportSettings:

Свойство	Описание
`Language DefaultLanguage`	Язык скрипта по умолчанию. Он будет использован для всех создаваемых отчетов.
`bool ShowProgress`	Определяет, надо ли показывать окно прогресса.
`bool ShowPerformance`	Определяет, надо ли показывать информацию о быстродействии (время построения отчета и занимаемая память) в окне просмотра.

Настройки, относящиеся к дизайнеру, доступны в свойстве Config.DesignerSettings:

Свойство	Описание
`Icon Icon`	Иконка окна дизайнера.
`Font DefaultFont`	Шрифт по умолчанию. Он будет использован для новых объектов "Текст".

Настройки, относящиеся к окну просмотра, доступны в свойстве Config.PreviewSettings:

Свойство	Описание
`PreviewButtons Buttons`	Набор кнопок, которые будут видны в окне просмотра.
`int PagesInCache`	Максимальное количество страниц готового отчета, которые хранятся в кэше в оперативной памяти.
`bool ShowInTaskbar`	Определяет, надо ли показывать окно в панели задач.
`bool TopMost`	Определяет, надо ли показывать окно над всеми остальными окнами.
`Icon Icon`	Иконка окна просмотра.
`string Text`	Текст в заголовке окна. Если это свойство оставить пустым, будет показан текст "ИмяОтчета - Предварительный просмотр".

Настройки, относящиеся к отсылке email (соответствующая кнопка есть в окне просмотра), доступны в свойстве Config.EmailSettings:

Свойство	Описание
`string Address`	Адрес отправителя (ваш адрес).
`string Name`	Имя отправителя (ваше имя).
`string MessageTemplate`	Шаблон письма, который будет использоваться при создании нового письма. Например, "Здравствуйте, С уважением, ...".
`string Host`	Адрес почтового сервера SMTP.
`int Port`	Порт SMTP (как правило, 25).
`string UserName, string Password`	Имя пользователя и пароль. Оставьте эти свойства пустыми, если почтовый сервер не требует аутентификацию.
`bool AllowUI`	Разрешает менять эти настройки в окне отправки сообщения. Настройки будут храниться в файле конфигурации FastReport.

Настройки интерфейса доступны в следующих свойствах:

Свойство	Описание
`UIStyle UIStyle`	Стиль интерфейса окна дизайнера и предварительного просмотра.

Кроме свойств, имеется несколько событий, которые позволяют:

- заменить диалоги "Открыть" и "Сохранить" в дизайнере;
- заменить окно прогресса;
- передать в отчет строку подключения.

Эти действия будут описаны далее.

Подмена диалогов "Открыть" и "Сохранить"

Если вы решили хранить отчеты в базе данных, вам может понадобиться изменить работу дизайнера так, чтобы вместо стандартных диалогов "Открыть файл" и "Сохранить файл" использовались ваши диалоги, работающие с базой данных. Для этого используйте глобальный класс FastReport.Utils.Config.DesignerSettings (см. раздел "Конфигурация среды FastReport"). Он имеет следующие события:

Событие `CustomOpenDialog`

Вызывается при показе диалога "Открыть файл". Вы должны показать собственный диалог в обработчике этого события. Если диалог был выполнен успешно, вы должны вернуть параметр e.Cancel = false, а так же вернуть имя файла в параметре e.FileName. Следующий пример показывает использование стандартного диалога открытия файла:

```
private void CustomOpenDialog_Handler(
  object sender, OpenSaveDialogEventArgs e)
{
  using (OpenFileDialog dialog = new OpenFileDialog())
  {
    dialog.Filter = "Report files (*.frx)|*.frx";
    // set e.Cancel to false if dialog
    // was succesfully executed
    e.Cancel = dialog.ShowDialog() != DialogResult.OK;
    // set e.FileName to the selected file name
    e.FileName = dialog.FileName;
  }
}
```

Событие `CustomSaveDialog`

Вызывается при показе диалога "Сохранить файл". Вы должны показать собственный диалог в обработчике этого события. Если диалог был выполнен успешно, вы должны вернуть параметр e.Cancel = false, а так же вернуть имя файла в параметре e.FileName. Следующий пример показывает использование стандартного диалога сохранения файла:

```
private void CustomSaveDialog_Handler(
  object sender, OpenSaveDialogEventArgs e)
{
  using (SaveFileDialog dialog = new SaveFileDialog())
  {
    dialog.Filter = "Report files (*.frx)|*.frx";
    // get default file name from e.FileName
    dialog.FileName = e.FileName;
    // set e.Cancel to false if dialog
    // was succesfully executed
    e.Cancel = dialog.ShowDialog() != DialogResult.OK;
    // set e.FileName to the selected file name
    e.FileName = dialog.FileName;
  }
}
```

Событие `CustomOpenReport`

Вызывается при чтении отчета. В обработчике события вы должны загрузить отчет e.Report из места,

указанного в параметре e.FileName. Параметр e.FileName содержит значение, которое вернул обработчик события CustomOpenDialog. Это может быть имя файла, имя записи в базе данных, и т.д. Следующий пример показывает, как загрузить отчет из файла:

```
private void CustomOpenReport_Handler(object sender, OpenSaveReportEventArgs e)
{
  // load the report from the given e.FileName
  e.Report.Load(e.FileName);
}
```

Событие `CustomSaveReport`

Вызывается при записи отчета. В обработчике события вы должны сохранить отчет e.Report в место, указанное в параметре e.FileName. Параметр e.FileName содержит значение, которое вернул обработчик события CustomSaveDialog. Это может быть имя файла, имя записи в базе данных, и т.д. Следующий пример показывает, как сохранить отчет в файл:

```
private void CustomSaveReport_Handler(object sender, OpenSaveReportEventArgs e)
{
  // save the report to the given e.FileName
  e.Report.Save(e.FileName);
}
```

Подмена стандартного окна прогресса

Окно прогресса показывается при выполнении следующих операций:

- построение отчета
- печать
- экспорт

Вы можете отключить окно прогресса, установив значение свойства FastReport.Utils.Config.ReportSettings.ShowProgress в false.

Вы также можете подключить собственное окно прогресса. Для этого используйте следующие события, которые определены в классе FastReport.Utils.Config.ReportSettings (см. раздел "Конфигурация среды FastReport"):

Событие	Описание
`StartProgress`	Вызывается один раз перед началом операции. В этом событии нужно создать свое окно прогресса и показать его.
`Progress`	Вызывается каждый раз после обработки очередной страницы отчета. В этом событии нужно показать текущее состояние прогресса.
`FinishProgress`	Вызывается один раз после окончания операции. В этом событии нужно разрушить свое окно прогресса.

В событие Progress передается параметр e типа ProgressEventArgs. Он имеет следующие свойства:

Тип, свойство	Описание
`string Message`	Текст сообщения.
`int Progress`	Номер текущей обрабатываемой страницы.
`int Total`	Общее количество страниц. Этот параметр может быть равен 0, если выполняется построение отчета, так как в этом случае общее количество страниц неизвестно.

Как правило, в обработчике события Progress достаточно показать текст, который передан в параметре e.Message, в собственном окне прогресса.

Передача в отчет строки подключения

Если ваш отчет использует собственные источники данных, вам может понадобиться передать в отчет строку подключения. Это можно сделать тремя способами.

Первый способ: строка подключения присваивается непосредственно объекту Connection в отчете:

```
report1.Load(...);
// делаем это после загрузки отчета, перед его построением
// подразумевается, что в отчете есть одно подключение
report1.Dictionary.Connections[0].ConnectionString = my_connection_string;
report1.Show();
```

Второй способ: строка подключения передается с помощью параметра отчета. Для этого сделайте следующее:

- запустите дизайнер отчета;
- в окне "Данные" создайте параметр отчета (например, с именем "MyParameter"). Подробнее об этом см. в "Руководстве пользователя";
- в окне "Данные" выделите объект "Подключение";
- перейдите в окно "Свойства" и установите значение свойства ConnectionStringExpression:

```
[MyParameter]
```

- передайте в отчет значение параметра MyParameter, содержащее строку подключения:

```
report1.SetParameterValue("MyParameter", my_connection_string);
```

Третий способ: используйте событие DatabaseLogin у глобального класса FastReport.Utils.Config.ReportSettings (см. раздел "Конфигурация среды FastReport"). Это событие вызывается каждый раз при открытии соединения с базой данных. Вот пример обработчика:

```
private void environmentSettings1_DatabaseLogin(
  object sender, DatabaseLoginEventArgs e)
{
  e.ConnectionString = my_connection_string;
}
```

> Учтите, что событие DatabaseLogin работает глобально для всех отчетов.

Передача в отчет текста SQL

Отчет может содержать источники данных, добавленные в "Мастере подключения к данным". Часто возникает необходимость передать в такой источник текст SQL, сформированный в приложении. Для этого используйте следующий код:

```
using FastReport.Data;
report1.Load(...);
// делаем это после загрузки отчета, перед его построением
// находим таблицу по ее имени (алиасу)
TableDataSource table = report1.GetDataSource("MyTable") as TableDataSource;
table.SelectCommand = "новый текст SQL";
report1.Show();
```

Обращение к объектам отчета

В случае, если вы храните отчет в виде класса (см. раздел "Хранение и загрузка отчета"), вы можете обращаться к объектам отчета напрямую. В примере ниже показано, как изменить шрифт объекта Text1:

```
SimpleListReport report = new SimpleListReport();
report.Text1.Font = new Font("Arial", 12);
```

В остальных случаях необходимо использовать метод FindObject объекта Report:

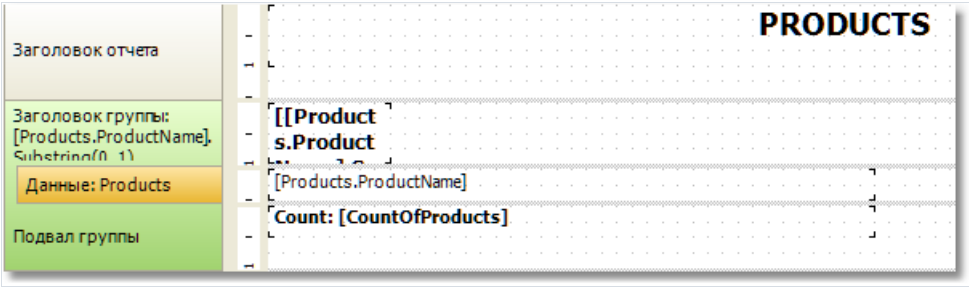
```
TextObject text1 = report1.FindObject("Text1") as TextObject;
text1.Font = new Font("Arial", 12);
```

Для обращения к источнику данных, определенному в отчете, используйте метод GetDataSource объекта Report. Этот метод принимает в качестве параметра алиас источника (т.е. его имя, которое видно в окне "Данные"):

```
DataSourceBase ds = report1.GetDataSource("Products");
```

Создание отчета с помощью кода

Рассмотрим пример, который создает отчет с группой следующего вида:




```
Report report = new Report();

// register the "Products" table
report.RegisterData(dataSet1.Tables["Products"], "Products");

// enable it to use in a report
report.GetDataSource("Products").Enabled = true;

// create A4 page with all margins set to 1cm
ReportPage page1 = new ReportPage();
page1.Name = "Page1";
report.Pages.Add(page1);

// create ReportTitle band
page1.ReportTitle = new ReportTitleBand();
page1.ReportTitle.Name = "ReportTitle1";

// set its height to 1.5cm
page1.ReportTitle.Height = Units.Centimeters * 1.5f;

// create group header
GroupHeaderBand group1 = new GroupHeaderBand();
group1.Name = "GroupHeader1";
group1.Height = Units.Centimeters * 1;

// set group condition
group1.Condition = "[Products.ProductName].Substring(0, 1)";

// add group to the page.Bands collection
page1.Bands.Add(group1);

// create group footer
group1.GroupFooter = new GroupFooterBand();
group1.GroupFooter.Name = "GroupFooter1";
group1.GroupFooter.Height = Units.Centimeters * 1;

// create DataBand
DataBand data1 = new DataBand();
data1.Name = "Data1";
data1.Height = Units.Centimeters * 0.5f;

// set data source
data1.DataSource = report.GetDataSource("Products");

// connect databand to a group
group1.Data = data1;

// create "Text" objects
// report title
```

```
// report title
TextObject text1 = new TextObject();
text1.Name = "Text1";

// set bounds
text1.Bounds = new RectangleF(0, 0, Units.Centimeters * 19, Units.Centimeters * 1);

// set text
text1.Text = "PRODUCTS";

// set appearance
text1.HorzAlign = HorzAlign.Center;
text1.Font = new Font("Tahoma", 14, FontStyle.Bold);

// add it to ReportTitle
page1.ReportTitle.Objects.Add(text1);

// group
TextObject text2 = new TextObject();
text2.Name = "Text2";
text2.Bounds = new RectangleF(0, 0, Units.Centimeters * 2, Units.Centimeters * 1);
text2.Text = "[[Products.ProductName].Substring(0, 1)]";
text2.Font = new Font("Tahoma", 10, FontStyle.Bold);

// add it to GroupHeader
group1.Objects.Add(text2);

// data band
TextObject text3 = new TextObject();
text3.Name = "Text3";
text3.Bounds = new RectangleF(0, 0, Units.Centimeters * 10, Units.Centimeters * 0.5f);
text3.Text = "[Products.ProductName]";
text3.Font = new Font("Tahoma", 8);

// add it to DataBand
data1.Objects.Add(text3);

// group footer
TextObject text4 = new TextObject();
text4.Name = "Text4";
text4.Bounds = new RectangleF(0, 0, Units.Centimeters * 10, Units.Centimeters * 0.5f);
text4.Text = "Count: [CountOfProducts]";
text4.Font = new Font("Tahoma", 8, FontStyle.Bold);

// add it to GroupFooter
group1.GroupFooter.Objects.Add(text4);

// add a total
Total groupTotal = new Total();
groupTotal.Name = "CountOfProducts";
groupTotal.TotalType = TotalType.Count;
groupTotal.Evaluator = data1;
groupTotal.PrintOn = group1.Footer;

// add it to report totals
report.Dictionary.Totals.Add(groupTotal);

// run the report
report.Show();
```

Готовый отчет выглядит следующим образом:

PRODUCTS

A

Aniseed Syrup
Alice Mutton
Count: 2

B

Boston Crab Meat
Count: 1

Использование собственного окна просмотра

Используя свойство FastReport.Utils.Config.PreviewSettings (см. раздел "Конфигурация среды FastReport"), вы можете настроить внешний вид и поведение стандартного окна просмотра.

Если вас по каким-то причинам не устраивает стандартное окно просмотра, вы можете создать свое собственное. Для этого используйте элемент управления PreviewControl (WpfPreviewControl для FastReport WPF), который можно добавить на форму. Для того чтобы показать отчет в собственном окне просмотра, подключите PreviewControl к отчету с помощью кода:

```
report1.Preview = previewControl1; // для FastReport .NET/Mono
report1.WpfPreview = previewControl1; // для FastReport WPF
```

Для построения отчета и отображения его в PreviewControl, используйте метод Show отчета:

```
report1.Show();
your_form.ShowDialog();
```

или следующий код:

```
if (report1.Prepare())
{
  report1.ShowPrepared();
  your_form.ShowDialog();
}
```

В этих примерах your_form - это ваша форма, на которой лежит PreviewControl.

Используя методы компонента PreviewControl, можно управлять его работой из кода. При этом можно отключить стандартную панель инструментов и/или строку статуса с помощью свойств ToolbarVisible, StatusbarVisible. В поставку FastReport .NET входит пример Demos\C#\CustomPreview, в котором показано, как управлять работой окна просмотра (для FastReport WPF пример находится в папке Demos\WPF\CustomPreview).

Фильтрация списка таблиц в "Мастере подключения"

Мастер подключения к данным может быть вызван из меню "Данные|Новый источник данных...". Здесь вы можете настроить подключение и выбрать одну или несколько таблиц данных. По умолчанию, мастер показывает все таблицы, доступные в подключении. Если вы хотите отфильтровать ненужные таблицы, используйте событие FastReport.Utils.Config.DesignerSettings.FilterConnectionTables. Следующий пример показывает, как убрать из списка таблицу с именем "Table 1":

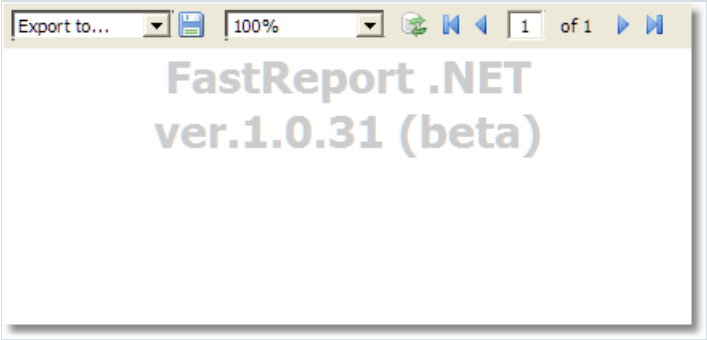
```
using FastReport.Utils;
Config.DesignerSettings.FilterConnectionTables += FilterConnectionTables;
private void FilterConnectionTables(
  object sender, FilterConnectionTablesEventArgs e)
{
  if (e.TableName == "Table 1")
    e.Skip = true;
}
```

Работа в ASP.NET

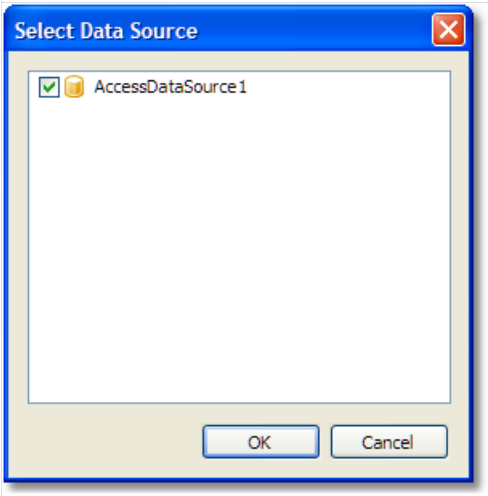
Использование компонента WebReport

Рассмотрим типичный сценарий использования компонента WebReport.

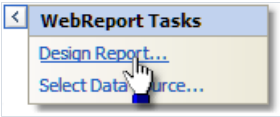
- предполагается, что вы имеете веб-проект, в котором настроены источники данных (например, AccessDataSource);
- положите компонент WebReport на вашу веб-форму:



- в меню "smart tag" выберите пункт "Select Data Source..." и выберите источник данных (один или несколько), который вы хотите использовать в отчете:



- в меню "smart tag" выберите пункт "Design Report..." для запуска дизайнера отчетов:



- создайте отчет. Подробнее об этом можно прочитать в "Руководстве пользователя";
- закройте дизайнер;
- сохраните изменения в вашем проекте и запустите его. Вы увидите окно с готовым отчетом.

Настройка обработчика

Для корректной работы WebReport требует настройки специального обработчика в файле web.config. При создании визуального объекта в среде Visual Studio необходимые строки автоматически прописываются в файл конфигурации. Компонент WebReport осуществляет проверку наличия нужной конфигурации на этапе исполнения приложения. В случае отсутствия нужных строк в файле web.config будет выдана ошибка с инструкцией по внесению изменений.

Файл web.config должен содержать следующие строки при условии использования совместно с сервером IIS6:

```
<system.web>
…
<httpHandlers>
<add path="FastReport.Export.axd" verb="*" type="FastReport.Web.Handlers.WebExport"/>
</httpHandlers>
…
</system.web>
```

При совместной работе с сервером IIS7 необходимы следующие строки:

```
<system.webServer>
…
<handlers>
<add name="FastReportHandler" path="FastReport.Export.axd" verb="*"
type="FastReport.Web.Handlers.WebExport"/>
</handlers>
…
</system.webServer>
```

При переносе проекта на другой сервер контролируйте наличие нужных строк конфигурации WebReport в файле web.config.

Проверить работоспособность обработчика WebReport можно, обратившись к следующему URL вашего приложения:

`http://yoursite/app_folder/FastReport.Export.axd`

Если обработчик настроен правильно, то вы увидите информационное сообщение о номере версии FastReport.

Хранение и загрузка отчета

Вы можете хранить отчет одним из следующих способов:

В коде веб-формы:

Типичный сценарий работы с отчетом, который мы рассмотрели выше, использует именно этот способ. Отчет хранится в свойстве ReportResourceString компонента WebReport. У данного способа есть следующие плюсы и минусы:

+ самый быстрый и простой способ работы с FastReport .NET;
- шаблон отчета хранится в ViewState веб-формы, что приведет к его пересылке на клиентскую машину. Это может замедлить работу, в случае, если отчет занимает много места;
- этот способ несовместим с режимом medium trust.

Загрузка отчета осуществляется автоматически.

В файле .FRX:

Этот способ предполагает хранение отчета в файле, в специальной папке "App_Data". Для этого:

- запустите дизайнер отчета;
- создайте отчет и сохраните его в файл .FRX;
- в окне "Solution Explorer", выберите папку "App_Data", щелкните на ней правой кнопкой мыши и выберите пункт "Add|Existing Item...". Выберите файл отчета, который вы только что сохранили;
- выберите компонент WebReport и очистите его свойство ReportResourceString;
- выберите свойство ReportFile и вызовите его редактор. В открывшемся окне выберите файл отчета из папки "App_Data".

У данного способа есть следующие плюсы и минусы:

+ файл отчета не передается на клиентскую машину;
- этот способ несовместим с режимом medium trust.

Загрузка отчета осуществляется автоматически.

Также возможна загрузка отчета из WebReport.StartReport. Пример кода обработчика StartReport:

```
(sender as WebReport).Report.Load(this.Server.MapPath("~/App_Data/report.frx"));
```

В виде класса C#/VB.NET:

Этот способ предполагает работу с классом отчета. Для этого:

- в дизайнере отчета выберите пункт "Файл|Сохранить как...", и выберите тип файла - "Файл C#" или "Файл VB.Net" (это зависит от того, какой язык выбран в самом отчете - см. "Руководство пользователя");
- полученный файл добавьте в ваш проект. Лучше всего хранить его в папке "App_Code";
- очистите свойства ReportResourceString и ReportFile компонента WebReport.

У данного способа есть следующие плюсы и минусы:

+ работа с отчетом, как с обычным классом;
+ доступны все возможности Visual Studio, в том числе отладка;
+ это единственный способ работы с отчетом, который совместим с режимом Medium Trust в ASP.NET;

- вы не можете редактировать такой отчет. Для этого нужно иметь оригинальный файл отчета в формате .FRX.

Для работы с отчетом создайте обработчик события WebReport.StartReport. В его коде сделайте следующее:

- создайте экземпляр класса отчета;
- зарегистрируйте данные;
- присвойте этот экземпляр свойству Report компонента WebReport.

Пример кода обработчика StartReport:

```
SimpleListReport report = new SimpleListReport();
report.RegisterData(your_data, "your_data_name");
WebReport1.Report = report;
```

Если необходимо показать ранее построенный отчет, то это также можно сделать, используя обработчик WebReport.StartReport и свойство WebReport.ReportDone. Пример кода обработчика StartReport для загрузки ранее подготовленного отчета:

```
(sender as WebReport).Report.LoadPrepared(this.Server.MapPath("~/App_Data/Prepared.fpx"));
(sender as WebReport).ReportDone = true;
```

Регистрация данных

В случае, если вы выбирали данные для отчета, используя "smart tag" и пункт меню "Select Data Source...", специальных действий для регистрации данных выполнять не нужно. В этом случае FastReport .NET хранит имена источников данных в свойстве ReportDataSources компонента WebReport. Если вы выбрали несколько источников данных, их имена разделяются запятыми.

В случае, если этот способ регистрации данных вам не подходит, используйте событие StartReport компонента WebReport. В этом событии вы можете вызвать методы RegisterData и RegisterDataAsp отчета. Отчет доступен в свойстве WebReport.Report:

```
webReport1.Report.RegisterData(myDataSet);
```

Подробнее о методах для регистрации данных можно прочитать в этом разделе.

Передача параметра в отчет

Для передачи значения параметра используется метод SetParameterValue объекта Report. Этот метод рассмотрен в главе "Работа в Windows.Forms".

Для передачи параметра в отчет используйте событие StartReport компонента WebReport. Отчет доступен в свойстве WebReport.Report:

```
webReport1.Report.SetParameterValue("MyParam", 10);
```

Работа в режиме Medium Trust

В этом режиме работает большинство shared-hosting провайдеров. Он накладывает следующие ограничения:

- невозможна компиляция кода отчета;
- невозможна работа с источниками данных MS Access;
- невозможно использование объекта RichObject;
- невозможно использование некоторых фильтров экспорта, которые делают вызовы WinAPI или создают временные файлы (PDF, Open Office);
- возможны другие ограничения, зависящие от конкретного провайдера.

Для работы с FastReport в этом режиме используйте способ хранения отчета в виде класса, как описано в разделе "Хранение и загрузка отчета". При этом компиляция отчета в процессе его работы не требуется.

Кроме того, необходимо, чтобы библиотека System.Windows.Forms.DataVisualization.dll была добавлена в GAC. Эта библиотека является частью Microsoft Chart Control и используется в FastReport для построения диаграмм. Проконсультируйтесь со своим shared-hosting провайдером по поводу добавления этой библиотеки в GAC.

Работа в архитектурах Web Farm и Web Garden

При использовании генератора отчетов FastReport в много-серверных (Web Farm) или многопроцессорных (Web Garden) архитектурах возникают дополнительные требования по созданию специального файлового хранилища для синхронизации данных между объектами WebReport.

Необходимо внести следующие строки в файл конфигурации web.config:

```
<appSettings>
    <add key="FastReportStoragePath" value="\\FS\WebReport_Exchange"/>
    <add key="FastReportStorageTimeout" value="10"/>
    <add key="FastReportStorageCleanup" value="1"/>
</appSettings>
```

FastReportStoragePath - путь к папке для хранения временных файлов, при работе в много-серверной архитектуре каждый из серверов должен иметь доступ к данной папке.

FastReportStorageTimeout - время для хранения кэша отчетов в минутах.

FastReportStorageCleanup - интервал в минутах для проверки просроченных элементов кэша отчетов.

Правильность конфигурации можно проконтролировать, обратившись к вашему приложению по URL:

`http://yoursite/app_folder/FastReport.Export.axd`

Результат должен содержать строку "Cluster mode: ON".

Работа в ASP.NET MVC

Использование FastReport в MVC разметке ASPX ничем не отличается от обычной работы с ASP.NET. Пример применения WebReport в разметке aspx можно посмотреть в папке \Demos\C#\MvcDemo.

На использовании WebReport в разметке Razor, которая появилась с версии MVC 3, нужно остановиться подробнее. Для корректной работы WebReport нужно добавить в файл web.config в корневой папке веб-приложения определения хендлеров. При использовании сервера IIS7 и выше нужно добавить следующую строку в секцию <system.webServer> :

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

При использовании IIS6 добавляется строка в секцию <system.web> :

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

Далее нужно внести изменения в файл web.config в папке, где находятся View. В секцию <system.web.webPages.razor> нужно добавить строки:

```
<add namespace="FastReport" />
<add namespace="FastReport.Web" />
```

Эти строки необходимы, чтобы можно было создавать объекты FastReport и обращаться к их свойствам непосредственно во View.

В файле _Layout.cshtml в теге добавляем строки:

```
@WebReportGlobals.Scripts()
@WebReportGlobals.Styles()
```

Теперь можно переходить к отображению отчета на View. Переходим в соответствующий контроллер и создаем там WebReport:

```
WebReport webReport = new WebReport(); // создаем объект
```

```
webReport.Width = 600;  // задаем ширину
webReport.Height = 800; // задаем высоту
webReport.Report.RegisterData(dataSet, "AppData"); // привязка источника данных
webReport.ReportFile = this.Server.MapPath("~/App_Data/report.frx");  // загрузка отчета из файла
ViewBag.WebReport = webReport; // передаем данные во View
```

В коде View добавляем строку в нужное место:

```
@ViewBag.WebReport.GetHtml()
```

Аналогичный код по созданию WebReport можно написать также непосредственно во View.

Пример использования WebReport в разметке Razor находится в папке \Demos\C#\MvcRazor. Там есть различные варианты загрузки отчета, в том числе и заранее подготовленного, а также есть пример использования события StartReport.

> Не забудьте добавить в папку bin каждого из примеров недостающие библиотеки.

Пример экспорта файла в MVC

При совместном использовании FastReport.Net с фреймворком ASP.Net MVC существует возможность простого формирования файла в нужном выходном формате при нажатии на кнопку формы.

Добавляем следующий код во View:

```
@using (Html.BeginForm("GetFile", "Home"))
{
        <input id="pdf" type="submit" value="Export to PDF" />
}
```

`GetFile` - имя обработчика результатов сабмита формы в контроллере, Home - имя контроллера, в данном случае файл контроллера `HomeController.cs`

Добавляем пространство имен в контроллере:

```
using FastReport.Export.Pdf;
```

Добавляем нужный метод в контроллер:

```
        public FileResult GetFile()
        {
            WebReport webReport = new WebReport();
            // привязка данных
            System.Data.DataSet dataSet = new System.Data.DataSet();
            dataSet.ReadXml(report_path + "nwind.xml");
            webReport.Report.RegisterData(dataSet, "NorthWind");

            // загружаем нужный файл отчета
            webReport.ReportFile = this.Server.MapPath("~/App_Data/report.frx");
            // строим отчет
            webReport.Report.Prepare();
            // сохраняем в нужном формате в поток
            Stream stream = new MemoryStream();
            webReport.Report.Export(new PDFExport(), stream);
            stream.Position = 0;
            // возвращаем результат в браузер
            return File(stream, "application/zip", "report.pdf");
        }
```

Привязка данных и загрузка файла отчета показаны для примера. Привязка к данным может быть выполнена непосредственно в самом файле отчета из дизайнера.

Если вам необходим другой формат, то нужно добавить соответствующий uses и вызвать нужный конструктор в строке экспорта. Например, для формата Excel 2007 нужны строки:

```
using FastReport.Export.OoXML;
...
webReport.Report.Export(new Excel2007Export(), stream);
...
return File(stream, "application/xlsx", "report.xlsx");
```

FastReport .NET и jQuery

Объект WebReport генератора отчетов FastReport .NET использует в своей работе библиотеку jQuery. Данная библиотека очень популярна и уже может быть использована в вашем проекте.

Чтобы избежать дублирования загрузки скриптов и стилей jQuery в браузер клиента при работе с разметкой Razor, нужно использовать следующие строки в файле _Layout.cshtml:

```
@WebReportGlobals.ScriptsWOjQuery()
@WebReportGlobals.StylesWOjQuery()
```

вместо стандартных, которые загружают все файлы

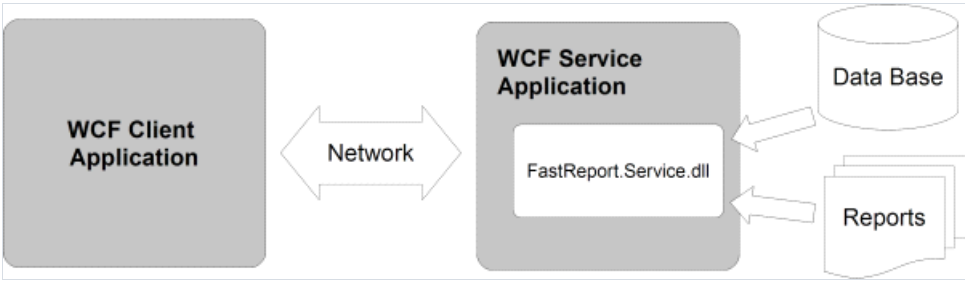
`@WebReportGlobals.Scripts()`
`@WebReportGlobals.Styles()`

При работе с разметкой ASPX нужно установить свойство ExternalJquery = true (по умолчанию false).

Работа с WCF

Библиотека FastReport.Service.dll

FastReport .NET содержит в своем составе библиотеку FastReport.Service.dll (только в сборке под .NET 4.0). Данная библиотека представляет собой WCF Service Library и предназначена для использования в составе пользовательских приложений, выполняющих функции сервисов.



В данный момент библиотека содержит следующие функции:

```
List<ReportItem> GetReportsList();
```

возвращает список доступных отчетов в виде элементов ReportItem. Отчеты хранятся на жестком диске, на сервере, где работает сервис. Передаются список всех отчетов с учетом иерархии папок. Файлы отсортированы по алфавиту.

```
List<ReportItem> GetReportsListByPath(string path);
```

возвращает список отчетов в виде элементов ReportItem из папки path. В список попадут все отчеты и подпапки. Файлы отсортированы по алфавиту.

`List<GearItem> GetGearList();`

возвращает список доступных форматов, которые может сформировать сервис отчетов в виде элементов GearItem.

`Stream GetReport(ReportItem report, GearItem gear);`

возвращает поток данных, который является результатом построения отчета report в формате gear. Параметры report и gear могут быть использованы из списков, полученных ранее, либо созданы новые объекты с необходимыми свойствами. Возвращаемый поток не поддерживает позиционирование - вы можете осуществлять из него только последовательное чтение.

Подробнее об элементах списков.

ReportItem

```
public class ReportItem
{
 public string Path;
 public string Name;
 public string Description;
 public Dictionary<string, string> Parameters;
}
```

Path – путь к файлу отчета на сервере, относительно корня папки хранения отчетов. Расширение файла отчета может быть только *.frx. Данное свойство используется для идентификации конкретного отчета при дальнейших запросах.

Name – имя отчета, берется из метаданных файла отчета. Если метаданные отчета содержат пустое название, то имя совпадает с именем файла отчета без расширения. Данное свойство может быть использовано для построения интерактивных списков доступных отчетов в вашем приложении (например, в ListBox).

Description – описание отчета, берется из метаданных файла отчета.

Dictionary<string, string> Parameters – словарь параметров отчета. Может быть заполен параметрами, которые будут впоследствии переданы в отчет. Поддерживаются только строковые значения, что необходимо учесть при разработке шаблона отчета.

GearItem

```
public class GearItem
{
 public string Name;
 public Dictionary<string, string> Properties;
}
```

Name – название формата. Может содержать одно из следующих строковых значений:

Name	Описание
PDF	Файл Adobe Acrobat
DOCX	Файл Microsoft Word 2007
XLSX	Файл Microsoft Excel 2007
PPTX	Файл Microsoft PowerPoint 2007
RTF	Файл Rich Text – поддерживается множеством текстовых редакторов
ODS	Файл Open Office Spreadsheet
ODT	Файл Open Office Text
MHT	Сжатый HTML файл вместе с изображениями, может быть открыт в Internet Explorer
CSV	Comma separated values - текстовая таблица со значениями, разделенными запятыми

Name	Описание
DBF	Файл базы данных dBase
XML	XML таблица Excel – изображения не поддерживаются
TXT	Текстовый файл
FPX	Формат готового отчета FastReport.Net, может быть загружен в программу Viewer.exe либо непосредственно в объект отчета из кода Report.LoadPrepared(stream) и показан на экране Report.ShowPrepared()

Dictionary<string, string> Properties – словарь параметров формирования отчета. Полный список поддерживаемых параметров со значениями по умолчанию доступен при запросе списка форматов от сервера.

При создании сервиса, использующего библиотеку FastReport.Service.dll нужно добавить настройки в файл App.config или Web.config вашего приложения.

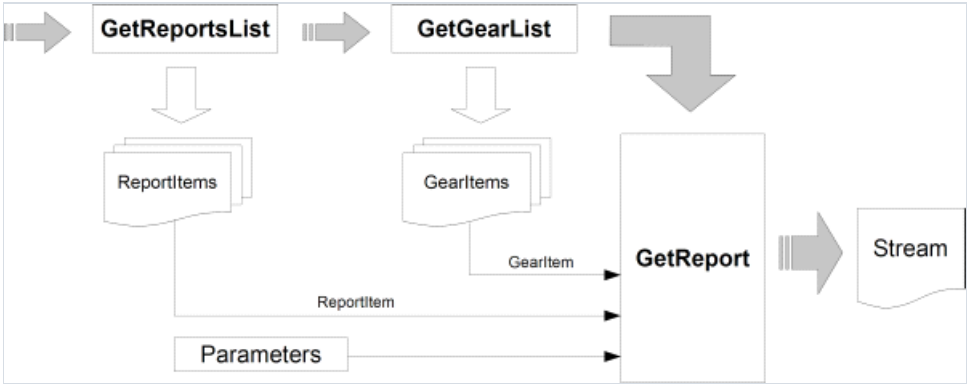
```
<appSettings>
<add key="FastReport.ReportsPath" value="C:\Program files\FastReports\FastReport.Net\Demos\WCF" />
<add key="FastReport.ConnectionStringName" value="FastReportDemo" />
<add key="FastReport.Gear" value="PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX" />
</appSettings>
```

FastReport.ReportsPath – указывает путь к папке с файлами отчетов, список которых будет передаваться клиенту.

FastReport.ConnectionStringName – имя строки подключения к базе данных, которое хранится в разделе файла конфигурации. Используется для замены внутренней строки подключения в шаблоне отчета.

FastReport.Gear – список доступных форматов. Можно выбрать только нужные и изменить порядок следования названий.

Схематичный вариант использования FastReport.Service:



Если вы точно знаете, какой отчет и в каком формате вы хотите получить, то схема использования сервиса может быть такой (это сократит количество запросов к сервису):

Важные моменты, которые необходимо учитывать при создании шаблонов отчетов для использования в сервисах:

- диалоги в отчетах не поддерживаются и будут пропущены при построении отчета;
- каждый отчет должен содержать внутренний DataConnection, строка подключения которого будет при построении отчета сервисом заменена на строку из конфигурации.

Примеры использования FastReport.Service.dll можно посмотреть в папках \Demos\C#\WCFWebService , \Demos\C#\WCFWindowsService , \Demos\C#\WCFWebClient , \Demos\C#\WCFClient. Пример файла конфигурации сервиса - FastReport.Service.dll.config.

Простой пример WCF сервиса

Данный пример не потребует программирования и предназначен для проверки работоспособности библиотеки и файла конфигурации. Для выполнения задачи мы используем программу WcfSvcHost.exe, которая идет в составе Visual Studio:

1. Создаем папку для наших экспериментов где-либо на диске, например C:\WCF\FastReport
2. Копируем в созданную папку файлы FastReport.Service.dll, FastReport.Service.dll.config, FastReport.dll, FastReport.Bars.dll.
3. Создаем две подпапки Data и Reports
4. В папку Data копируем файл базы из примеров \FastReport.Net\Demos\Reports\nwind.xml
5. В папку Reports копируем содержимое папки \FastReports\FastReport.Net\Demos\WCF – она содержит тестовые отчеты со встроенными подключениями к базе данных, что является необходимым требованием при их использовании с библиотекой FastReport.Service.dll
6. Открываем файл конфигурации FastReport.Service.dll.config в любом текстовом редакторе.
7. Изменяем путь к папке с отчетами в секции

```
<add key="FastReport.ReportsPath" value="C:\WCF\FastReport\Reports" />
```

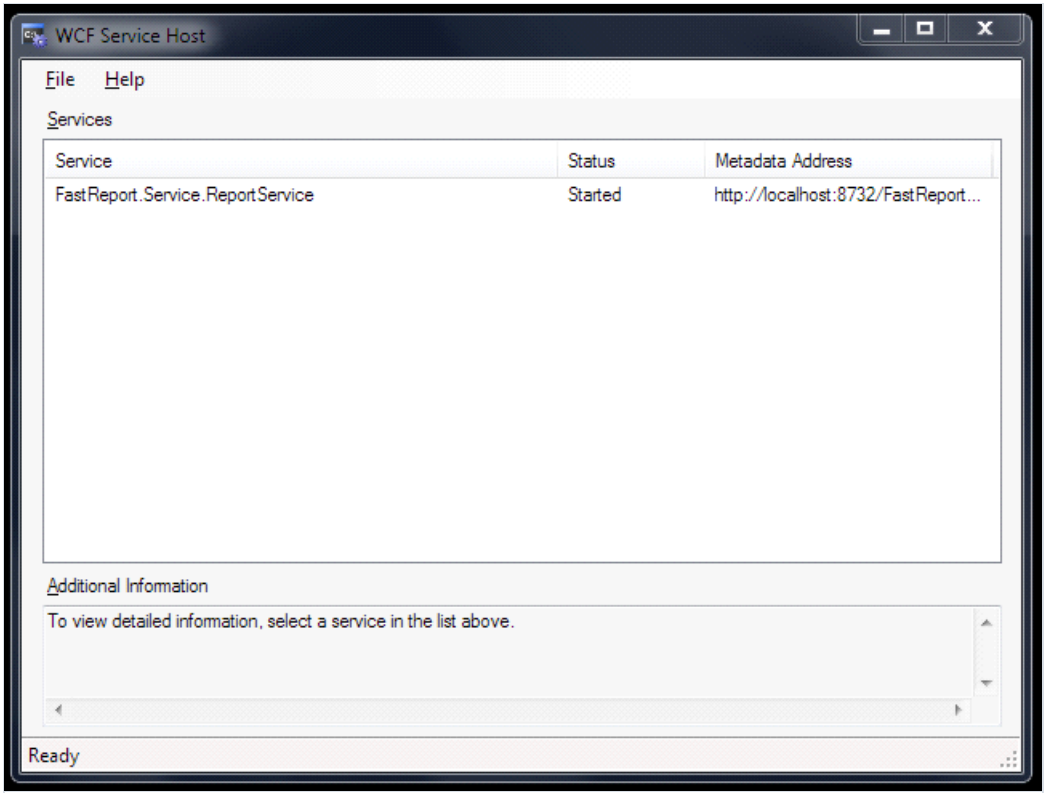
8. В секции меняем строку подключения к нашей базе:

```
<add name="FastReportDemo" connectionString="XsdFile=;XmlFile=C:\WCF\FastReport\Data\nwind.xml"/>
```

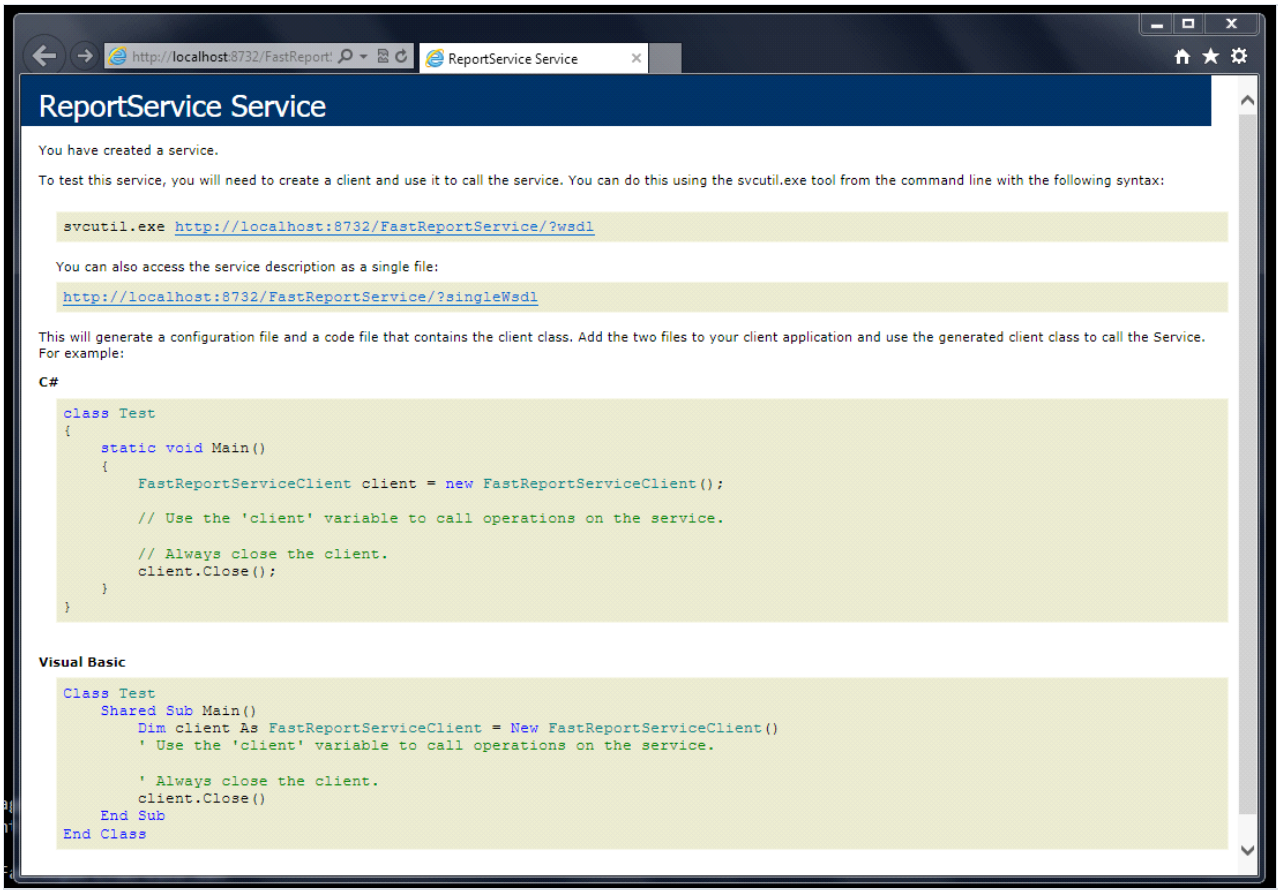
9. Создаем файл service.bat со следующей строкой:

"C:\Program Files\Microsoft Visual Studio 10.0\Common7\IDE\WcfSvcHost.exe" /service:C:\WCF\FastReport\FastReport.Service.dll /config:C:\WCF\FastReport\FastReport.Service.dll.config

10. Запускаем service.bat из проводника с правами администратора (Run as administrator). В трее появится иконка WCF Service Host. По двойному клику по ней должно открыться окно со следующим содержимым:



11. Откроем браузер и перейдем по адресу: http://localhost:8732/FastReportService/

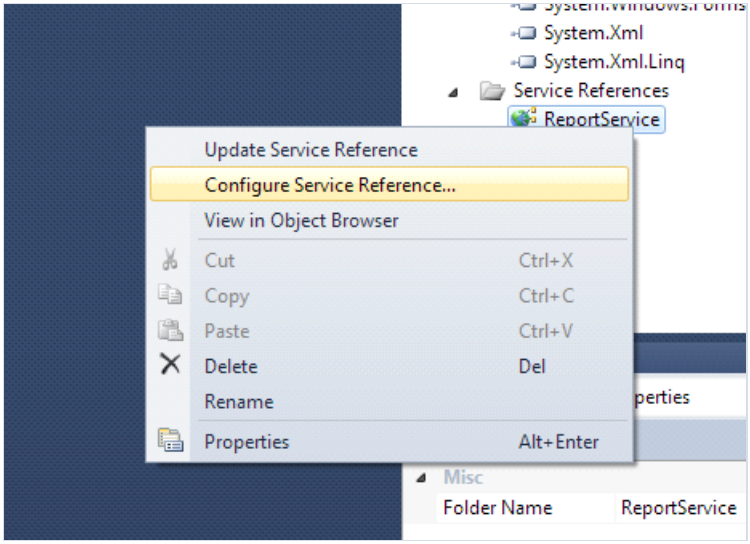


Номер порта сервиса можно настроить в файле конфигурации в строке

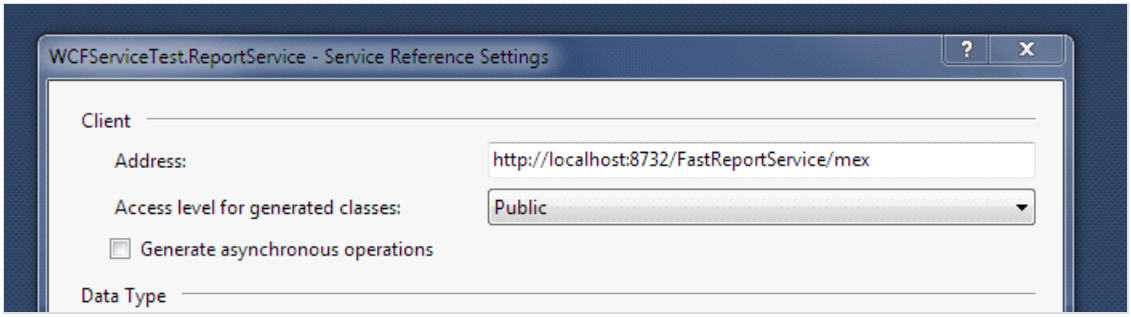
```
<add baseAddress="http://localhost:8732/FastReportService/" />
```

Подключиться к сервису можно из примера \FastReport.Net\Demos\C#\WCFClient

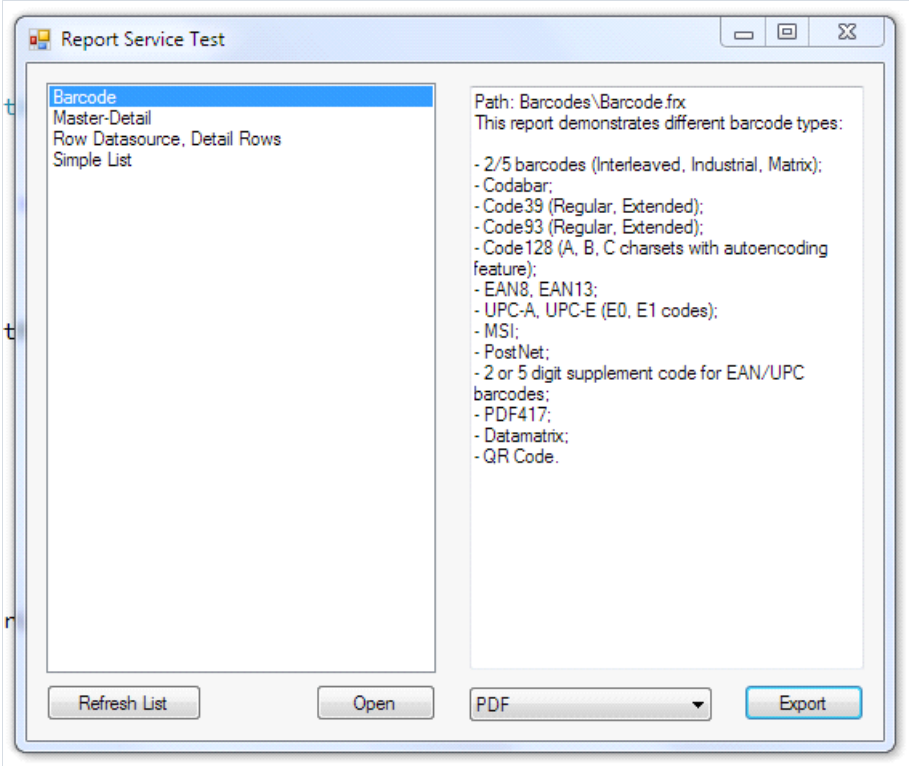
1. Открываем WCFServiceClient.csproj в Visual Studio
2. В дереве Solution Explorer кликаем правой кнопкой на Service References – ReportService и выбираем в меню Configure Service Reference



3. В открывшемся окне уточняем адрес нашего сервиса. В конце адреса нужно добавить строку "/mex" (metadata exchange)



4. Компилируем и запускаем пример.

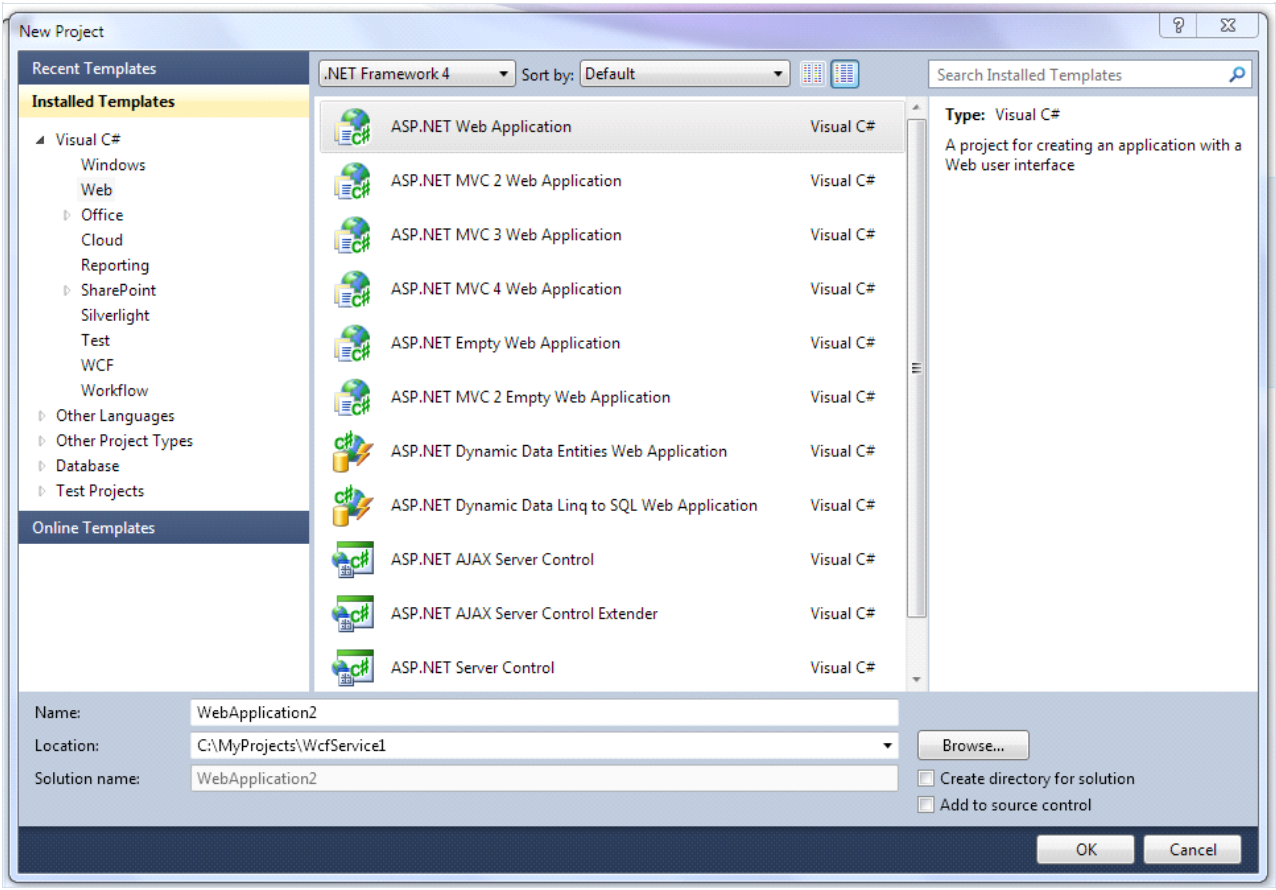

Report Service Test
Barcode
Master-Detail
Row Datasource, Detail Rows
Simple List
Path: Barcodes\Barcode.frx
This report demonstrates different barcode types:
- 2/5 barcodes (Interleaved, Industrial, Matrix);
- Codabar;
- Code39 (Regular, Extended);
- Code93 (Regular, Extended);
- Code128 (A, B, C charsets with autoencoding feature);
- EAN8, EAN13;
- UPC-A, UPC-E (E0, E1 codes);
- MSI;
- PostNet;
- 2 or 5 digit supplement code for EAN/UPC barcodes;
- PDF417;
- Datamatrix;
- QR Code.
Refresh List
Open
PDF
Export

Создание веб-сервиса на основе FastReport.Service.dll

Можно очень просто реализовать веб-сервис, используя библиотеку FastReport.Service.dll (WCF Service Library), которая поставляется совместно с FastReport.

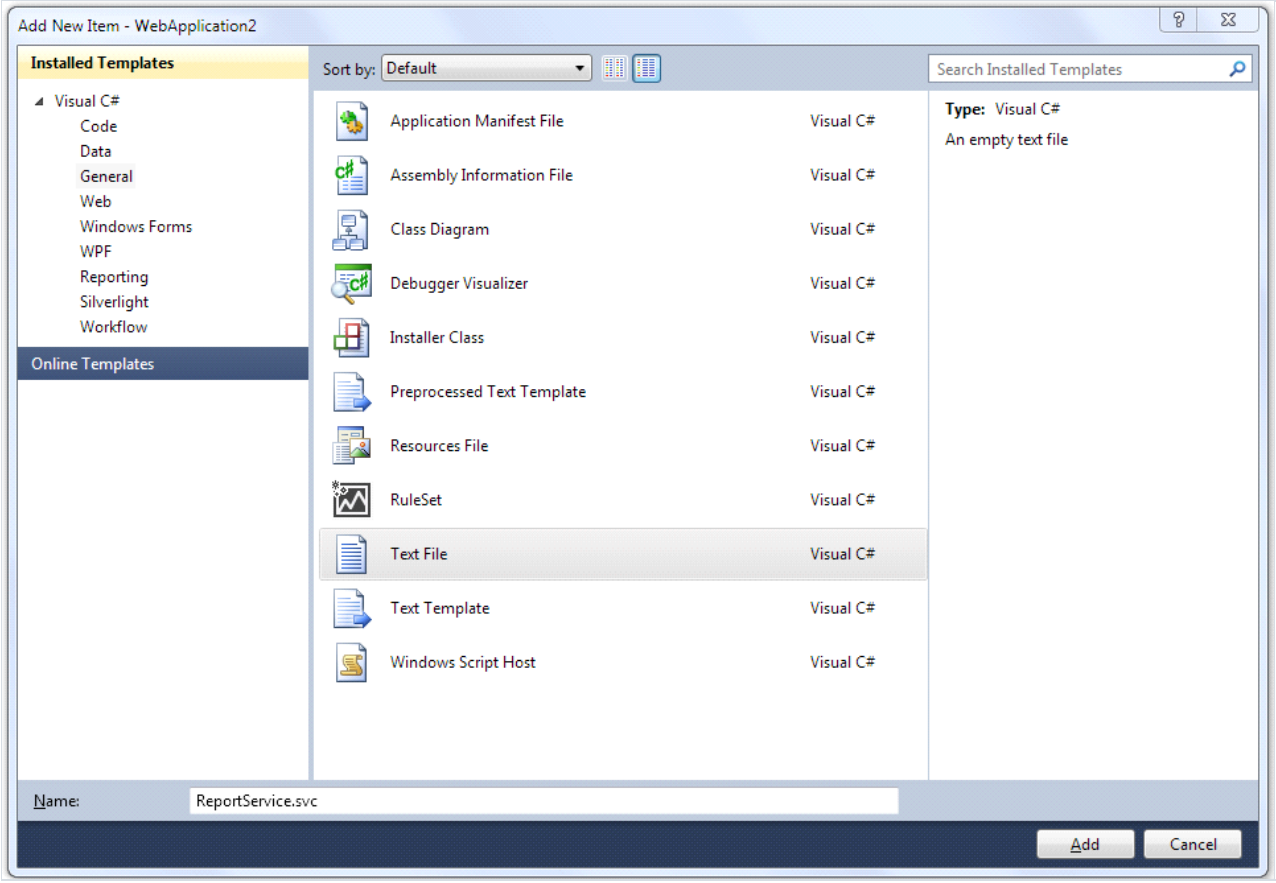
Пример основан на создании простейшего веб-приложения с функциями веб-сервиса с нуля, но так же можно доработать уже существующий ваш проект – главное, чтобы он работал под управлением .NET Framework 4.0 или более новым.

Запускаем Visual Studio и создаем новый проект ASP.NET Web Application под .NET Framework 4.0.



Добавляем reference на библиотеки FastReport.dll, FastReport.Bars.dll, FastReport.Service.dll

Создаем текстовый файл с именем ReportService.svc в корне сайта.



Добавляем следующие строки в созданный файл:

```
<%@ ServiceHost Service="FastReport.Service.ReportService" %>
<%@ Assembly Name="FastReport.Service" %>
```

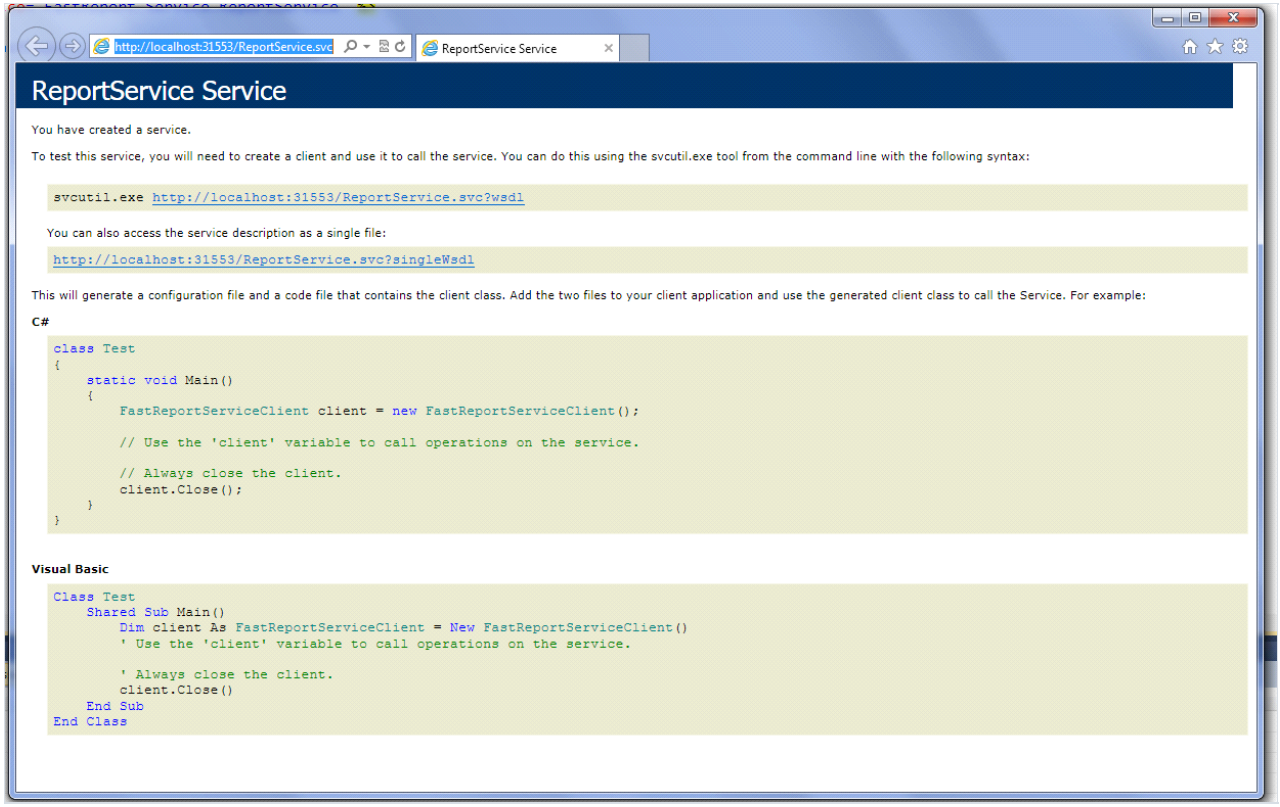
Открываем файл web.config и добавляем следующие секции в секцию :

```
<appSettings>
 <!-- path to folder with reports -->
 <add key="FastReport.ReportsPath" value="C:\Program files\FastReports\FastReport.Net\Demos\WCF" />
 <!-- name of connection string for reports -->
 <add key="FastReport.ConnectionStringName" value="FastReportDemo" />
 <!-- Comma-separated list of available formats PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX.
 You can delete any or change order in this list. -->
 <add key="FastReport.Gear" value="PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX" />
 </appSettings>
 <connectionStrings>
 <add name="FastReportDemo" connectionString="XsdFile=;XmlFile=C:\Program
Files\FastReports\FastReport.Net\Demos\Reports\nwind.xml"/>
 </connectionStrings>
 <system.serviceModel>
 <services>
 <service behaviorConfiguration="FastReportServiceBehavior" name="FastReport.Service.ReportService">
 <endpoint address="" binding="wsHttpBinding" contract="FastReport.Service.IFastReportService">
 <identity>
 <dns value="localhost" />
 </identity>
 </endpoint>
 <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
 </service>
 </services>
 <behaviors>
 <serviceBehaviors>
 <behavior name="FastReportServiceBehavior">
 <serviceMetadata httpGetEnabled="True" />
 <serviceDebug includeExceptionDetailInFaults="True" />
 </behavior>
 </serviceBehaviors>
 </behaviors>
 <bindings>
 <basicHttpBinding>
 <binding messageEncoding="Mtom"
 closeTimeout="00:02:00" openTimeout="00:02:00"
 receiveTimeout="00:10:00" sendTimeout="00:02:00"
 maxReceivedMessageSize="67108864" maxBufferSize="65536"
 transferMode="Streamed">
 <security mode="None">
 <transport clientCredentialType="None" />
 </security>
 </binding>
 </basicHttpBinding>
 </bindings>
 </system.serviceModel>
```

Параметр "FastReport.ReportsPath" должен указывать на папку с отчетами – для примера можно указать папку с примерами отчетов «\FastReport.Net\Demos\WCF».

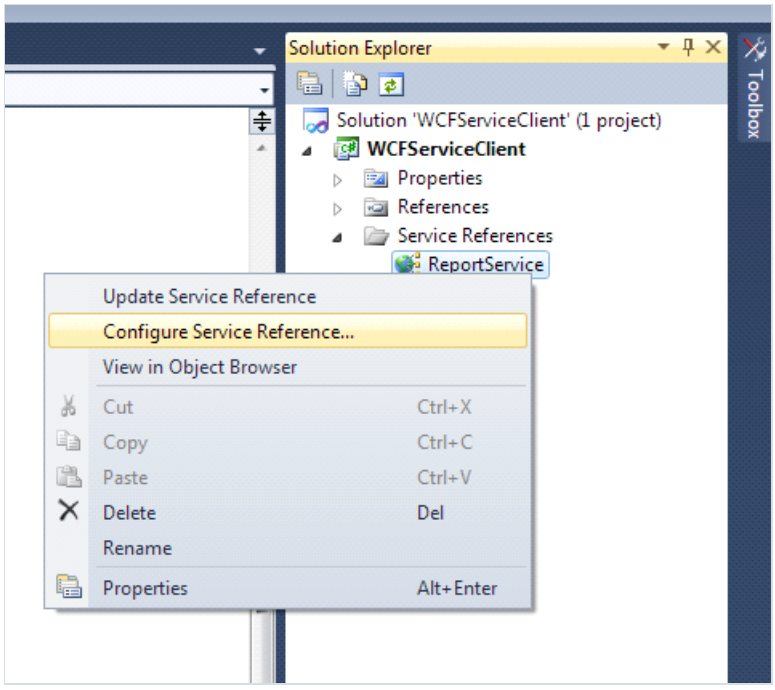
Параметр "FastReport.ConnectionStringName" содержит название строки подключения к базе данных. Эта строка должна быть прописана в секции .

Запускаем сайт на исполнение и проверяем доступность веб-сервиса обратившись к файлу ReportService.svc, который размещен на сайте.



При размещении проекта на сервере не забудьте проверить наличие файлов FastReport.dll, FastReport.Bars.dll, FastReport.Service.dll в папке /bin.

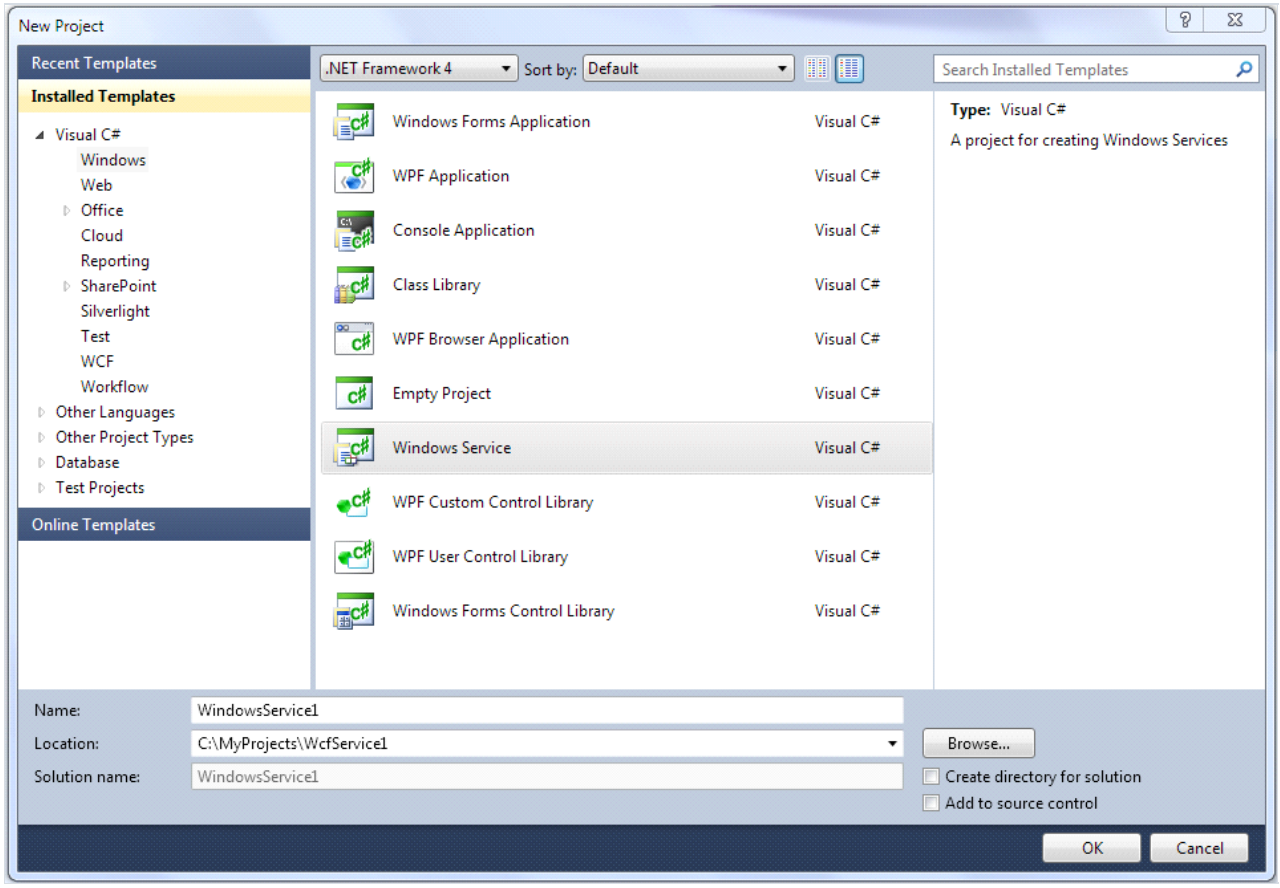
Примеры клиентских программ можно посмотреть в папках \FastReport.Net\Demos\C#\WCFClient и \FastReport.Net\Demos\C#\WCFWebClient. В каждом из примеров нужно сделать настройку Service References. Откройте проект в Visual Studio и по клику правой кнопкой мыши на ReportService выберете пункт меню Configure Service Reference.



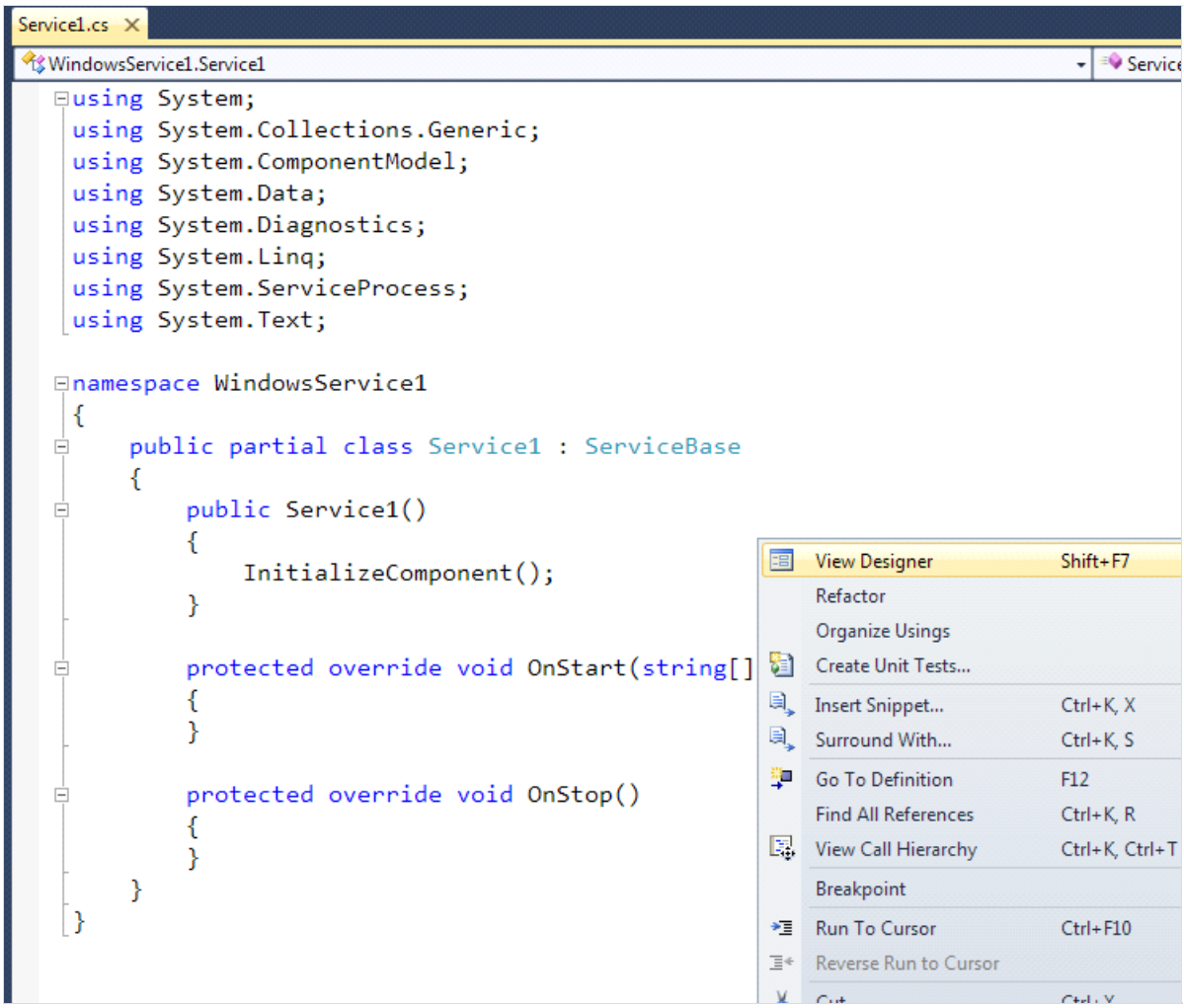
В открывшемся окне укажите адрес работающего веб-сервиса.

Создание сервиса WCF на основе службы Windows

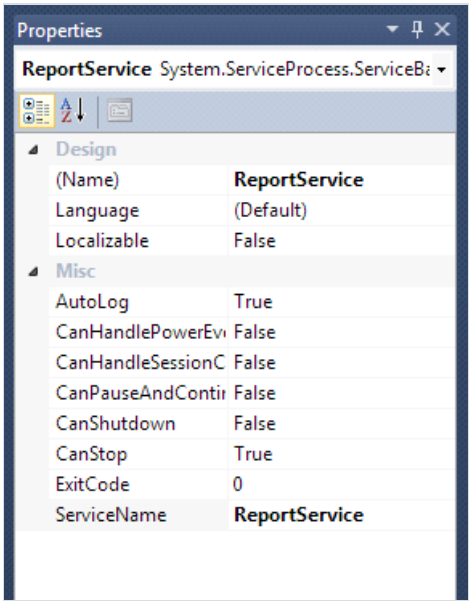
Нужно открыть Visual Studio и создать проект WindowsService.



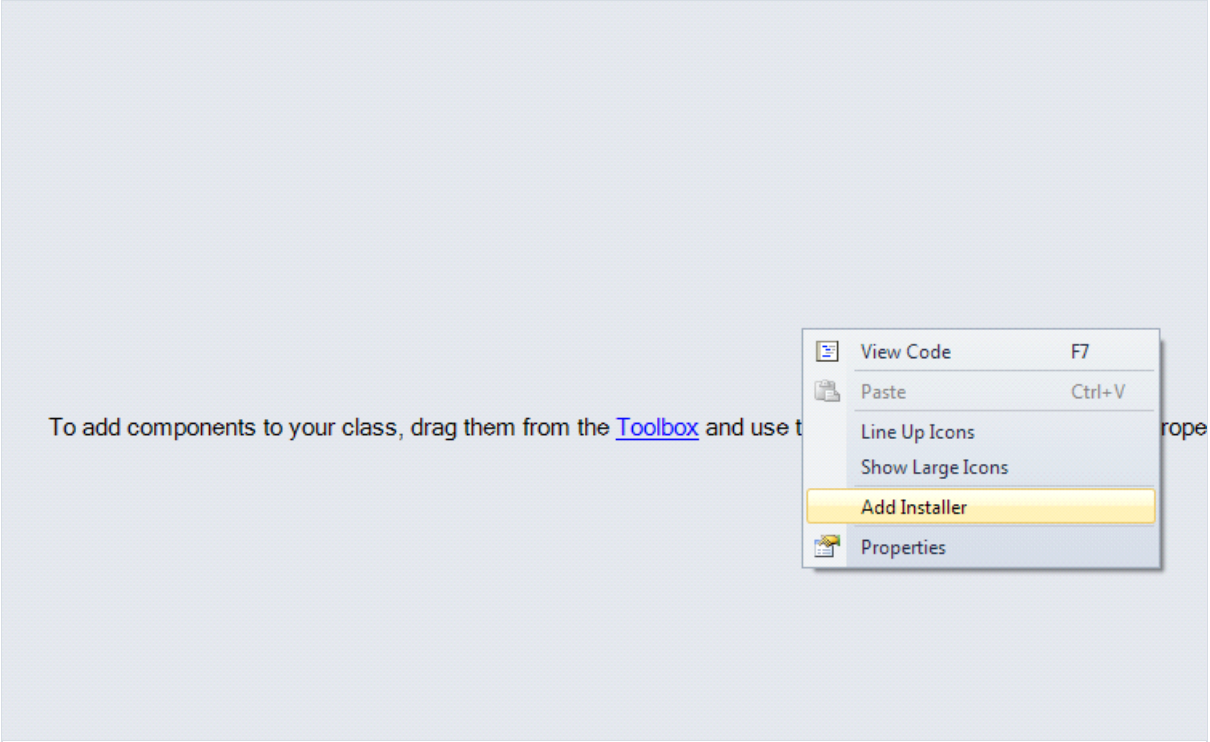
Открываем дизайнер Service1.cs



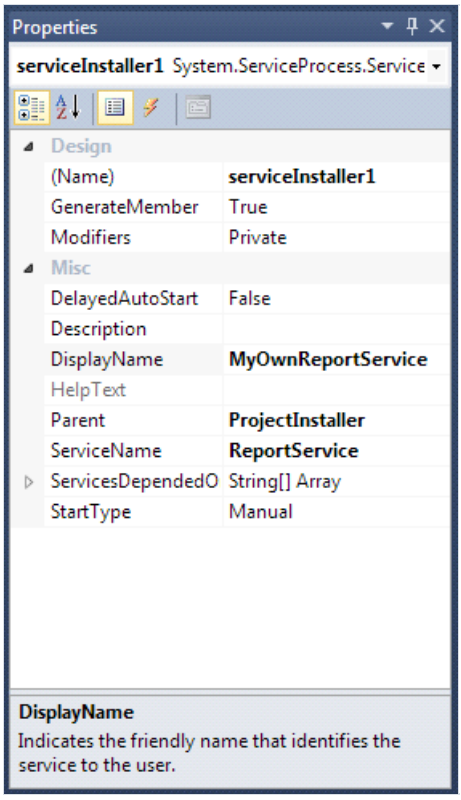
Изменяем имя сервиса по умолчанию на свое.



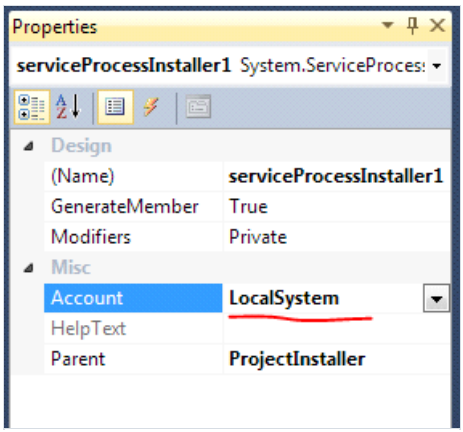
Кликаем по пустому окну дизайнера и выбираем пункт меню Add Installer.



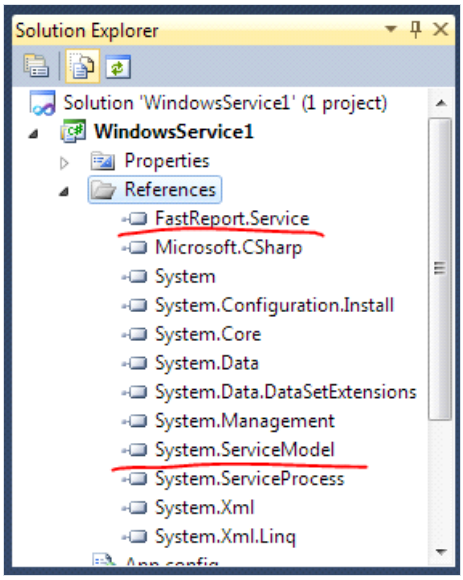
Редактируем свойства компонента serviceInstaller1 – указываем нужный DisplayName.



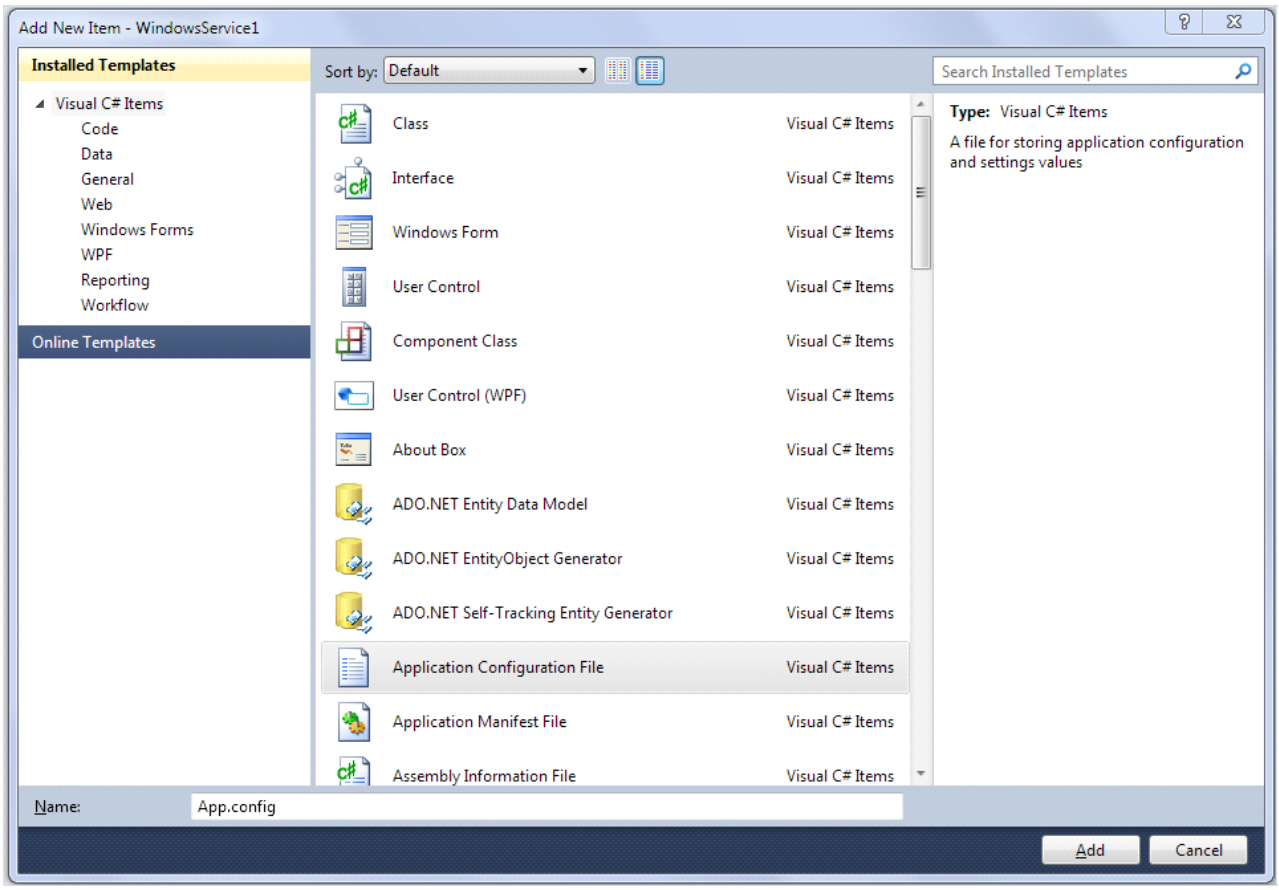
В свойствах компонента serviceProcessInstaller1 указываем тип аккаунта для работы службы LocalSystem.



Добавляем reference на System.ServiceModel и FastReport.Service.dll



Создаем конфигурационный файл службы.



Копируем следующий текст в созданный app.config:

```
<?xml version="1.0"?>
<configuration>
 <appSettings>
 <!-- path to folder with reports -->
 <add key="FastReport.ReportsPath" value="C:\Program files\FastReports\FastReport.Net\Demos\WCF" />
 <!-- name of connection string for reports -->
 <add key="FastReport.ConnectionStringName" value="FastReportDemo" />
 <!-- Comma-separated list of available formats PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX.
 You can delete any or change order in this list. -->
 <add key="FastReport.Gear" value="PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX" />
 </appSettings>
 <connectionStrings>
 <add name="FastReportDemo" connectionString="XsdFile=;XmlFile=C:\Program
Files\FastReports\FastReport.Net\Demos\Reports\nwind.xml"/>
 </connectionStrings>
 <system.web>
 <compilation debug="true" />
 <membership defaultProvider="ClientAuthenticationMembershipProvider">
 <providers>
 <add name="ClientAuthenticationMembershipProvider"
type="System.Web.ClientServices.Providers.ClientFormsAuthenticationMembershipProvider,
System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri=""
/>
 </providers>
 </membership>
 <roleManager defaultProvider="ClientRoleProvider" enabled="true">
 <providers>
 <add name="ClientRoleProvider" type="System.Web.ClientServices.Providers.ClientRoleProvider,
System.Web.Extensions, Version=4.0.0.0, Culture=neutral, PublicKeyToken=31bf3856ad364e35" serviceUri=""
cacheTimeout="86400" />
 </providers>
 </roleManager>
 </system.web>
 <!-- When deploying the service library project, the content of the config file must be added to the
host's
 app.config file. System.Configuration does not support config files for libraries. -->
 <system.serviceModel>
 <services>
 <service behaviorConfiguration="FastReportServiceBehavior" name="FastReport.Service.ReportService">
 <endpoint address="" binding="wsHttpBinding" contract="FastReport.Service.IFastReportService">
 <identity>
 <dns value="localhost" />
 </identity>
 </endpoint>
 <endpoint address="mex" binding="mexHttpBinding" contract="IMetadataExchange" />
 <host>
 <baseAddresses>
 <add baseAddress="http://localhost:8732/FastReportService/" />
 </baseAddresses>
 </host>
 </service>
 </services>
 <behaviors>
 <serviceBehaviors>
 <behavior name="FastReportServiceBehavior">
 <serviceMetadata httpGetEnabled="True" />
 <serviceDebug includeExceptionDetailInFaults="True" />
 </behavior>
 </serviceBehaviors>
 </behaviors>
 <bindings>
 <basicHttpBinding>
 <binding messageEncoding="Mtom"
 closeTimeout="00:02:00" openTimeout="00:02:00"
 receiveTimeout="00:10:00" sendTimeout="00:02:00"
```

```
 maxReceivedMessageSize="67108864" maxBufferSize="65536"
 transferMode="Streamed">
 <security mode="None">
 <transport clientCredentialType="None" />
 </security>
 </binding>
 </basicHttpBinding>
 </bindings>
 </system.serviceModel>
 <startup>
 <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0" />
 </startup>
</configuration>
```

Переходим в редактор файла Service1.cs. Добавляем строку:

```
using System.ServiceModel;
```

Видоизменяем класс службы, чтобы он выглядел следующим образом:

```
public partial class ReportService : ServiceBase
{
  ServiceHost reportHost;

  public ReportService()
  {
    InitializeComponent();
  }

  protected override void OnStart(string[] args)
  {
    if (reportHost != null)
      reportHost.Close();
    reportHost = new ServiceHost(typeof(FastReport.Service.ReportService));
    reportHost.Open();
  }

  protected override void OnStop()
  {
    reportHost.Close();
    reportHost = null;
  }
}
```

Установить полученную службу можно с помощью консольной утилиты InstallUtil.exe, которая поставляется в .NET Framework, например:

C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe
"C:\MyProjects\WcfService1\WindowsService1\bin\Debug\WindowsService1.exe"

Запустить службу на исполнение можно командой

net start ReportService

Можно открыть браузер и перейти по адресу http://localhost:8732/FastReportService/, который был указан в файле app.config, в параметре baseAddress. При желании вы можете изменить папку и порт.

Команды для останова службы и удаления:

net stop ReportService

C:\Windows\Microsoft.NET\Framework\v4.0.30319\InstallUtil.exe /u
"C:\MyProjects\WcfService1\WindowsService1\bin\Debug\WindowsService1.exe"

Аналогичный пример службы можно посмотреть в комплекте поставки FastReport .NET в папке
"\Demos\C#\WCFWindowsService"

Работа с Blazor

В вашем распоряжении есть пакет `FastReport.Web` с компонентами для встраивания в ваше Blazor Server веб-приложение. Начиная с версии 2021.3 данная библиотека содержит Razor компоненты для Blazor Server (server-side) приложения и это означает, что все операции будут выполняться на стороне сервера, который будет затем передавать клиенту готовое отображение.

Минимальная целевая платформа (target framework) на данный момент .Net Core 3.1 для обеспечения максимально возможной совместимости с последней на данный момент LTS (long-term support) версией с длительной поддержкой. Также данная версия есть у большинства пользователей и она совместима с последним фреймворком .NET 5 (в рамках данного пакета).

Настройки перед использованием

Для использования Blazor компонентов в пакете `FastReport.Web` вам необходимо добавить зависимость в ваш файл проекта (csproj) PackageReference с указанием id данного пакета и пакета `FastReport.Core` (версии могут отличаться):

```
<ItemGroup>
  <PackageReference Include="FastReport.Core" Version="2021.3.0-demo"/>
  <PackageReference Include="FastReport.Web" Version="2021.3.0-demo"/>
</ItemGroup>
```

Затем, для упрощения наименования, советуем добавить следующие namespace в импорты вашего проекта (файл `_Imports.razor`):

```
@using FastReport.Web
@using FastReport.Web.Blazor.Components
@using FastReport.Web.Blazor.Components.Internal
```

На самом деле, хватит только добавления `FastReport.Web.Blazor.Components`, однако, для некоторых случаев вам, возможно, понадобятся и другие namespace. Также некоторые компоненты, вероятно, будут перемещаться в рамках этих namespace во время стадии beta.

В конфигураторе вашего веб приложения необходимо вызвать метод `UseFastReport` с необязательным лямбда выражением настройки `FastReportOptions`. Также, для работы некоторых встроенных общих стилей и svg изображений иконок в Toolbar и Tab, необходимо использовать вызов `UseStaticFiles` (если вы не собираетесь использовать Toolbar и Tabs, вызов `UseStaticFiles` для использования данного пакета необязателен):

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    // ...
    app.UseStaticFiles();
    // ...
    app.UseFastReport();
    // ...
}
```

Описание встроенных компонентов

Далее идёт описание компонентов, входящих в состав `FastReport.Web`. Мы не рекомендуем использование компонентов, отличных от `WebReportContainer`, т.к. они могут вести себя нестабильно вне стандартного дерева компонентов.

Однако, для более точной настройки отображения, вам может потребоваться использование других компонентов.

WebReportContainer

Основной и самый универсальный Blazor компонент, выполняющий отрисовку `WebReport`, является `<WebReportContainer/>`. Он находится в namespace `FastReport.Web.Blazor.Components`. Единственный параметр, который он принимает - объект класса `WebReport`. Это означает, что для использования данного компонента, вы должны создать объект класса `WebReport`, присвоить ему отчёт (Report), другие необходимые параметры и передать данный объект в параметры `WebReportContainer`.

Пример

```
<WebReportContainer WebReport="@UserWebReport" />

@code {
    public WebReport UserWebReport { get; set; }

    protected override void OnParametersSet()
    {
        var report = Report.FromFile(
        Path.Combine(
            directory,
            "My report.frx"));

        // Registers the application dataset
        Report.RegisterData(DataSet, "NorthWind");

        UserWebReport = new WebReport();
        UserWebReport.Report = Report;
    }
}
```

Данный компонент умеет определять различный Mode (Designer, Dialog и обычный Preview). Умеет подготавливать отчет, встраивать стили, определенные по умолчанию, и индивидуальные стили, отображать Toolbar, Outline и Tabs, работать с интерактивными отчетами и др.

WebReportPreview

Аналогичен предыдущему компоненту, однако не учитывает Designer Mode. Т.е. всегда пытается подготовить и отрисовать отчёт.

ReportContainer

Аналогичен предыдущему компоненту, однако не включает загрузку стилей `WebReport` (общих и индивидуальных для `Toolbar/Tabs/Outline`).

Занимается подготовкой отчёта и последующим отображением вместе с Toolbar и Tabs (по необходимости).

При работе какой-либо интерактивности (клики по отчёту/работа с диалоговыми формами) обновляется именно этот компонент.

ReportBody / ExportComponent

`ReportBody` вызывает отрисовку Outline (по необходимости) и "вкладывает" в себя компонент, который является отрисовкой самого отчёта (`ExportComponent`), который ему передаёт `ReportContainer` . *Не рекомендуется для использования.*

BlazorExport

Самым "нижним" уровнем компонента является вовсе не компонент, а сам `BlazorExport` - инструмент экспорта подготовленного отчёта в формат построения RenderTreeBuilder. Находится в `FastReport.Web.Blazor.Export` namespace.

Для построения данного экспорта необходимо:

1. Подготовить данный отчёт
2. Быть уверенным, что в этом отчёте не используются диалоговые формы (они отображаются с помощью `DialogPageComponent` и не рассматриваются в данном руководстве)
3. Создать свой компонент и в нём явно определить способ построения (вызвать переопределение метода `BuildRenderTree`)
4. В этом методе построения создать экземпляр `BlazorExport` , задать ему необходимые свойства и вызвать Export с передачей следующих параметров: отчёта (Report) и экземпляр builder, являющимся аргументом данного переопределенного метода.

```
        /// Main function
        protected override void BuildRenderTree(RenderTreeBuilder builder)
        {
            using (BlazorExport blazor = new BlazorExport())
            {
                blazor.StylePrefix = $"fr{WebReport.ID}";
                blazor.EmbedPictures = true;
                blazor.OnClick += ProcessClick;
                blazor.EnableMargins = WebReport.EnableMargins;
                blazor.SinglePage = true;
                blazor.CurPage = WebReport.CurrentPageIndex;

                blazor.Export(myReport, builder);
            }
        }
```

Online Designer

В данный момент Online Designer может работать в элементе iframe средствами javascript и он полностью совместим со сборкой Online Designer для Core.

Для использования только возможностей дизайнера, вы можете вызвать компонент `<IFrameDesigner/>` с передачей ему параметра `WebReport` с настроенным свойством Report и необязательными `DesignerLocale` и `DesignerPath` :

```
    <IFrameDesigner WebReport="CurrentWebReport" />
```

Однако напоминаем, что компонент WebReportContainer понимает, с каким Mode он в данный момент

работает и вовсе не обязательно вызывать IFrameDesigner именно в таком виде.

Настройка общих стилей и SVG

В отличие от `FastReport.Web for Core`, SVG изображения для Toolbar и Tabs, а также некоторые общие стили отображения Tabs, Outline и др. были вынесены в `staticWebAssets` для возможной настройки в вашем веб-приложении (изменения цвета, размеров, замены изображения).

Располагаются данные ресурсы на вашем локальном хранилище. В момент разработки/сборки вашего приложения они находятся по адресу: `{UserName}/.nuget/packages/fastreport.web/{version}/staticwebassets`

В момент публикации вашего веб-приложения (dotnet publish) эти ресурсы копируются в директорию: `wwwroot/_content/FastReport.Web`

Демо проект

Проект для демонстрации работы с пакетом FastReport.Web вы можете найти на нашем GitHub. Пример использования `WebReportContainer` находится в пользовательском компоненте по пути `Pages/Index.razor` и `Pages/Index.razor.cs`.

Расширение функциональности FastReport

Создание собственных типов подключения

Подключения (data connection) используются для добавления источника данных в отчет. Это позволяет подключаться к данным прямо из отчета, а не использовать данные, которые предоставляет приложение.

Для создания своего подключения вам необходимо выполнить следующее:

- создать класс подключения - наследник FastReport.Data.DataConnectionBase и реализовать часть его методов;
- создать редактор подключения - наследник FastReport.Data.ConnectionEditors.ConnectionEditorBase и реализовать пользовательский интерфейс и методы GetConnectionString, SetConnectionString;
- зарегистрировать подключение в FastReport.

Эти шаги будут рассмотрены ниже.

Базовый класс подключения

Для создания собственного подключения используйте класс FastReport.Data.DataConnectionBase. Этот класс имеет следующий набор методов, которые нужно перекрыть при создании собственного подключения:

```
public abstract class DataConnectionBase : DataComponentBase
{
  protected virtual string GetConnectionStringWithLoginInfo(string userName, string password)
  public abstract string QuoteIdentifier(string value, DbConnection connection);
  public virtual DbConnection GetConnection();
  public virtual DbDataAdapter GetAdapter(string selectCommand, DbConnection connection,
    CommandParameterCollection parameters);
  public virtual ConnectionEditorBase GetEditor();
  public virtual Type GetParameterType();
  public virtual string GetConnectionId();
  public virtual string[] GetTableNames();
  public virtual void TestConnection();
  public virtual void FillTableSchema(DataTable table, string selectCommand,
    CommandParameterCollection parameters);
  public virtual void FillTableData(DataTable table, string selectCommand,
    CommandParameterCollection parameters);
}
```

Перекрывать требуется не все методы - некоторые из них имеют реализацию по умолчанию, которой будет достаточно для большинства случаев. Так, если ваше подключение использует объекты типа DbConnection и DbDataAdapter (а это стандарт для подавляющего большинства подключений), то вам потребуется реализовать следующие методы:

```
  public abstract string QuoteIdentifier(string value, DbConnection connection);
  protected virtual string GetConnectionStringWithLoginInfo(string userName, string password)
  public virtual DbConnection GetConnection();
  public virtual DbDataAdapter GetAdapter(string selectCommand, DbConnection connection,
    CommandParameterCollection parameters);
  public virtual ConnectionEditorBase GetEditor();
  public virtual Type GetParameterType();
  public virtual string GetConnectionId();
  public virtual string[] GetTableNames();
```

Если ваше подключение не работает с такими объектами, а, например, представляет собой мост между сервером приложений и отчетом, вам потребуется реализовать следующие методы:

```
  public abstract string QuoteIdentifier(string value, DbConnection connection);
  public virtual ConnectionEditorBase GetEditor();
  public virtual Type GetParameterType();
  public virtual string GetConnectionId();
  public virtual string[] GetTableNames();
  public virtual void TestConnection();
  public virtual void FillTableSchema(DataTable table, string selectCommand,
    CommandParameterCollection parameters);
  public virtual void FillTableData(DataTable table, string selectCommand,
    CommandParameterCollection parameters);
```

Ниже приведено описание каждого метода.

Свойство ConnectionString

Каждый тип подключения имеет свой набор параметров, например, путь к базе данных, диалект SQL, имя пользователя и пароль. Для хранения параметров используется свойство ConnectionString, которое имеет строковый тип.

Это свойство используется следующим образом:

- в методе GetConnection при создании объекта типа DbConnection;
- в редакторе подключения. В последнем случае из строки подключения выбираются отдельные параметры (например, путь к файлу БД). Для этого используется класс типа DbConnectionStringBuilder, специфичный для каждого типа подключения. Например, для MS SQL это SqlConnectionStringBuilder.

Метод QuoteIdentifier

В этот метод передается идентификатор (имя таблицы или колонки). Метод должен вернуть заключенный в кавычки идентификатор. Символы кавычек специфичны для каждой СУБД. Например, в MS Access используются квадратные скобки:

```
select * from [Table with long name]
```

соответственно, данный метод в MsAccessDataConnection выглядит следующим образом:

```
public override string QuoteIdentifier(string value, DbConnection connection)
{
  return "[" + value + "]";
}
```

Вы должны перекрыть этот метод. Обратитесь к руководству по данному типу подключения, чтобы узнать, какие символы используются для обращения к таблицам с длинными именами.

Метод GetConnectionStringWithLoginInfo

Этот метод должен взять существующую строку подключения (из свойства ConnectionString), включить в нее информацию об имени пользователя и пароле, и вернуть модифицированную строку. Этот метод используется, если в настройках подключения указано "Спрашивать пароль при подключении".

Например, реализация данного метода для MsSqlDataConnection выглядит так:

```
protected override string GetConnectionStringWithLoginInfo(string userName, string password)
{
  // берем существующую строку подключения
  SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder(ConnectionString);

  // включаем в нее имя пользователя и пароль
  builder.IntegratedSecurity = false;
  builder.UserID = userName;
  builder.Password = password;

  // возвращаем модифицированную строку
  return builder.ToString();
}
```

Вы должны перекрыть этот метод.

Метод GetConnection

Метод возвращает новый объект типа DbConnection, специфичный для данного подключения. Этот объект используется для соединения с БД, когда таблицу требуется заполнить данными.

Параметры подключения берутся из свойства ConnectionString. Например, для MsSqlDataConnection метод выглядит следующим образом:

```
public override DbConnection GetConnection()
{
  return new SqlConnection(ConnectionString);
}
```

Чаще всего вы должны перекрыть этот метод. Он используется в двух других методах - FillTableSchema и FillTableData. Если ваш тип подключения не использует объекты DbConnection и DbDataAdapter для получения информации о таблице и загрузки в нее данных, вы можете не перекрывать данный метод. В этом случае необходимо перекрыть методы FillTableSchema и FillTableData.

Метод GetAdapter

Метод возвращает новый объект типа DbDataAdapter, специфичный для данного подключения. Этот объект используется для заполнения таблицы данными.

В метод передаются следующие параметры:

Параметр	Описание
selectCommand	Текст запроса на языке SQL.
connection	Объект типа DbConnection, созданный в методе GetConnection.
parameters	Параметры запроса, если они определены.

Рассмотрим пример реализации данного метода в MsSqlDataConnection:

```
public override DbDataAdapter GetAdapter(string selectCommand, DbConnection connection,
  CommandParameterCollection parameters)
{
  SqlDataAdapter adapter = new SqlDataAdapter(selectCommand, connection as SqlConnection);
  foreach (CommandParameter p in parameters)
  {
    SqlParameter parameter = adapter.SelectCommand.Parameters.Add(p.Name, (SqlDbType)p.DataType, p.Size);
    parameter.Value = p.Value;
  }
  return adapter;
}
```

Метод создает адаптер, специфичный для MS SQL, заполняет параметры запроса и возвращает адаптер. На параметрах следует остановиться более подробно. В тексте запроса могут содержаться параметры. В этом случае в метод передается коллекция параметров в переменной parameters - это параметры, определенные в дизайнере FastReport при создании запроса. Вы должны добавить параметры в список adapter.SelectCommand.Parameters. Каждый параметр в коллекции parameters имеет следующие свойства:

Свойство	Описание
Name	Имя параметра.
DataType	Тип данных параметра. Это свойство типа int; вы должны привести его к типу, который используется для параметров в данном подключении.
Size	Размер данных параметра.
Value	Значение параметра.

Чаще всего вы должны перекрыть этот метод. Он используется в двух других методах - FillTableSchema и FillTableData. Если ваш тип подключения не использует объекты DbConnection и DbDataAdapter для получения информации о таблице и загрузки в нее данных, вы можете не перекрывать данный метод. В этом случае необходимо перекрыть методы FillTableSchema и FillTableData.

Метод GetEditor

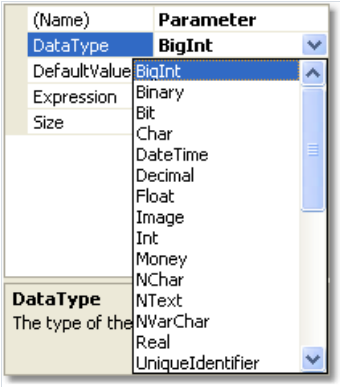
Этот метод возвращает экземпляр редактора для данного типа подключения. Реализация редактора будет рассмотрена в разделе "Редактор подключения".

Пример реализации данного метода в MsSqlDataConnection:

```
public override ConnectionEditorBase GetEditor()
{
  return new MsSqlConnectionEditor();
}
```

Метод GetParameterType

Значение, возвращаемое данным методом, используется в редакторе параметров запроса, для выбора типа данных параметра:



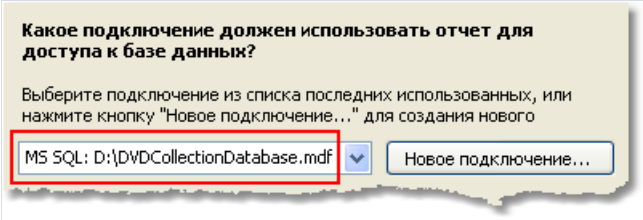
Например, для MsSqlDataConnection параметр имеет тип SqlDbType:

```
public override Type GetParameterType()
{
  return typeof(SqlDbType);
}
```

Вы должны перекрыть этот метод.

Метод GetConnectionId

Значение, возвращаемое этим методом, используется в "Мастере подключения" для отображения короткой информации по подключению:



Это значение, как правило, содержит название типа подключения и имя базы данных. В MsSqlDataConnection реализация метода выглядит следующим образом:

```
public override string GetConnectionId()
{
  SqlConnectionStringBuilder builder = new SqlConnectionStringBuilder(ConnectionString);
  string info = builder.InitialCatalog;
  if (String.IsNullOrEmpty(info))
    info = builder.AttachDBFilename;
  return "MS SQL: " + info;
}
```

Обратите внимание, что для разбора строки подключения (свойство ConnectionString) используется SqlConnectionStringBuilder.

Вы должны перекрыть этот метод.

Метод GetTableNames

Этот метод возвращает список таблиц и представлений (view), имеющихся в БД. Для этого, как правило, используется метод GetSchema объекта DbConnection, который возвращается методом GetConnection.

Данный метод имеет стандартную реализацию. В случае, если стандартная реализация несовместима с вашим типом подключения (т.е. не возвращает список сущностей БД), вам придется перекрыть этот метод. Рассмотрим в качестве примера реализацию этого метода в MsSqlDataConnection:

```
public override string[] GetTableNames()
{
  List<string> list = new List<string>();
  GetDBObjectNames("BASE TABLE", list);
  GetDBObjectNames("VIEW", list);
  return list.ToArray();
}

private void GetDBObjectNames(string name, List<string> list)
{
  DataTable schema = null;
  using (DbConnection connection = GetConnection())
  {
    connection.Open();
    schema = connection.GetSchema("Tables", newstring[] { null, null, null, name });
  }
  foreach (DataRow row in schema.Rows)
  {
    list.Add(row["TABLE_NAME"].ToString());
  }
}
```

Метод TestConnection

Этот метод вызывается, когда в настройках подключения вы нажимаете кнопку "Тест".

Метод имеет реализацию по умолчанию, которая создает подключение и пытается открыть его:

```
public virtual void TestConnection()
{
  DbConnection conn = GetConnection();
  if (conn != null)
  {
    try
    {
      conn.Open();
    }
    finally
    {
      conn.Dispose();
    }
  }
}
```

Если тест прошел успешно, метод ничего не делает. Иначе при открытии подключения возникает exception, который обрабатывается в "Мастере подключения" и выдает окно с текстом ошибки.

Как правило, вам не надо перекрывать этот метод. Это может понадобиться в случае, если ваше подключение не использует объект DbConnection для доступа к БД (и, соответственно, вы не возвращаете его в методе GetConnection).

Метод FillTableSchema

Метод заполняет схему таблицы (т.е. имена полей и их типы).

В метод передаются следующие параметры:

Параметр	Описание
table	Объект DataTable, схему которого необходимо заполнить.
selectCommand	Текст запроса на языке SQL.
parameters	Параметры запроса, если они определены.

Метод имеет реализацию по умолчанию, которая использует объекты, возвращаемые методами GetConnection и GetAdapter:

```
public virtual void FillTableSchema(DataTable table, string selectCommand, CommandParameterCollection
parameters)
{
 using (DbConnection conn = GetConnection())
 {
  OpenConnection(conn);

  // prepare select command
  selectCommand = PrepareSelectCommand(selectCommand, table.TableName, conn);

  // read the table schema
  using (DbDataAdapter adapter = GetAdapter(selectCommand, conn, parameters))
  {
   adapter.SelectCommand.CommandTimeout = CommandTimeout;
   adapter.FillSchema(table, SchemaType.Source);
  }
 }
}
```

В большинстве случаев вам не нужно перекрывать этот метод. Это может понадобиться, если вы не используете объекты DbConnection и DbDataAdapter для доступа к данным (и, соответственно, не реализуете методы GetConnection и GetAdapter).

Метод FillTableData

Метод заполняет таблицу данными.

В метод передаются следующие параметры:

Параметр	Описание
table	Объект DataTable, который необходимо заполнить.
selectCommand	Текст запроса на языке SQL.
parameters	Параметры запроса, если они определены.

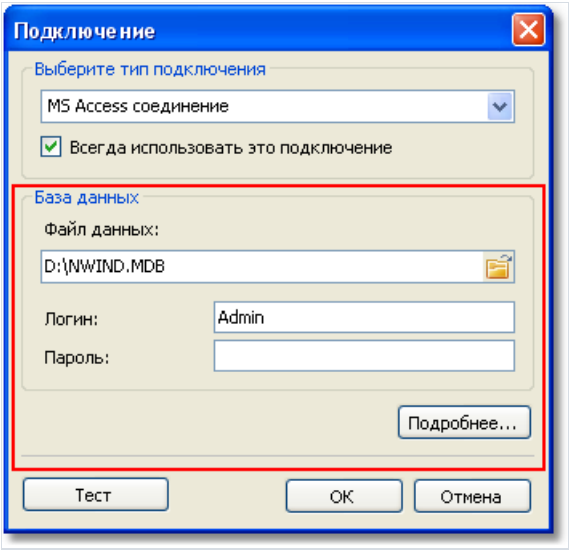
Метод имеет реализацию по умолчанию, которая использует объекты, возвращаемые методами GetConnection и GetAdapter:

```
public virtual void FillTableData(DataTable table, string selectCommand, CommandParameterCollection
parameters)
{
 using (DbConnection conn = GetConnection())
 {
  OpenConnection(conn);

  // prepare select command
  selectCommand = PrepareSelectCommand(selectCommand, table.TableName, conn);

  // read the table
  using (DbDataAdapter adapter = GetAdapter(selectCommand, conn, parameters))
  {
   adapter.SelectCommand.CommandTimeout = CommandTimeout;
   table.Clear();
   adapter.Fill(table);
  }
 }
}
```

В большинстве случаев вам не нужно перекрывать этот метод. Это может понадобиться, если вы не используете объекты DbConnection и DbDataAdapter для доступа к данным (и, соответственно, не реализуете методы GetConnection и GetAdapter).

Редактор подключения

Для редактирования подключения используется элемент управления типа UserControl, который отображается в окне выбора подключения (на рисунке элемент выделен красной рамкой):

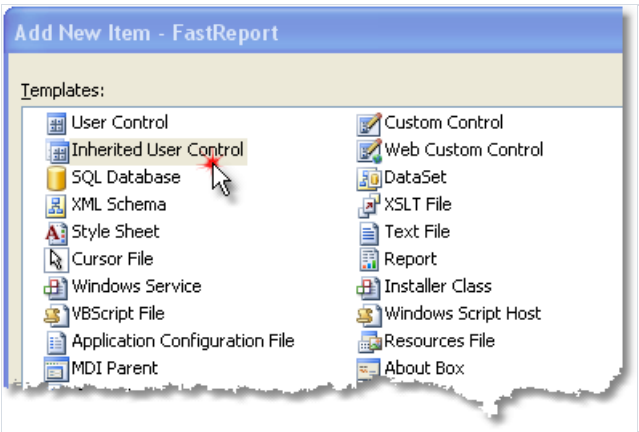


Для каждого типа подключения имеется свой редактор. Все редакторы наследуются от базового класса - FastReport.Data.ConnectionEditors.ConnectionEditorBase:

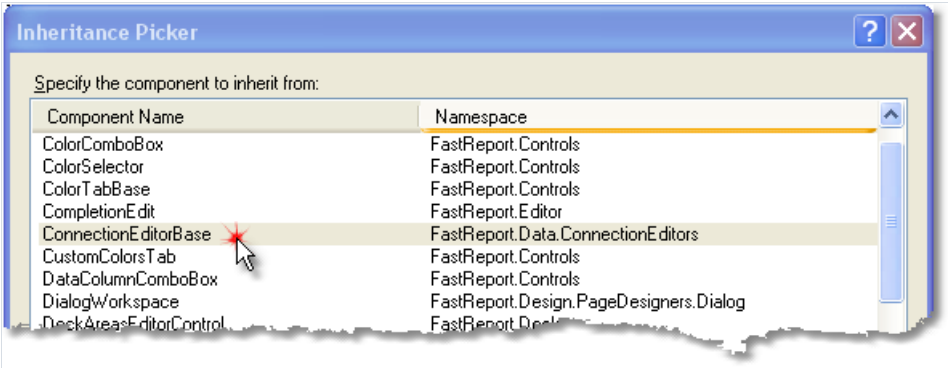
```
public class ConnectionEditorBase : UserControl
{
  protected virtual string GetConnectionString();
  protected virtual void SetConnectionString(string value);
}
```

Для создания своего редактора сделайте следующее:

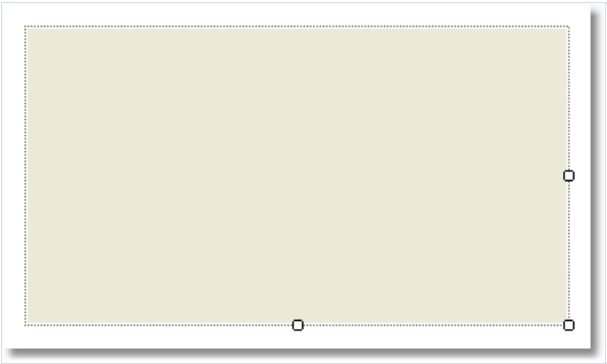
- добавьте в проект новый элемент управления (Add->User Control...) и выберите в окне "Add New Item" элемент "Inherited User Control":



- выберите тип базового класса - ConnectionEditorBase:



В проект будет добавлен элемент управления, который выглядит следующим образом:



На поверхности редактора разместите необходимые вам элементы управления. Измените высоту редактора, чтобы вместить все элементы. Ширина редактора фиксированная, изменять ее нельзя.

Методы базового класса позволяют заполнить элементы управления значениями из подключения и, наоборот, передать значения из элементов в подключение.

Метод GetConnectionString

Этот метод вызывается при закрытии окна редактора кнопкой "OK". Он должен вернуть строку подключения, которая содержит значения, введенные в редакторе.

Метод SetConnectionString

Этот метод вызывается при первом отображении редактора. Он должен разобрать строку подключения на составные части и отобразить их значения в редакторе. Для разбора строки подключения удобно использовать класс типа DbConnectionStringBuilder, который имеется в каждом типе подключения. Например, для MS SQL это SqlConnectionStringBuilder.

Регистрация подключения в FastReport

Для того чтобы ваше подключение можно было использовать в дизайнере FastReport, его необходимо зарегистрировать. Это можно сделать двумя способами.

Способ 1. Ваше подключение является частью вашего проекта (т.е. не вынесено в отдельный .dll модуль). Регистрация выполняется с помощью следующего кода:

```
FastReport.Utils.RegisteredObjects.AddConnection(typeof(MyDataConnection));
```

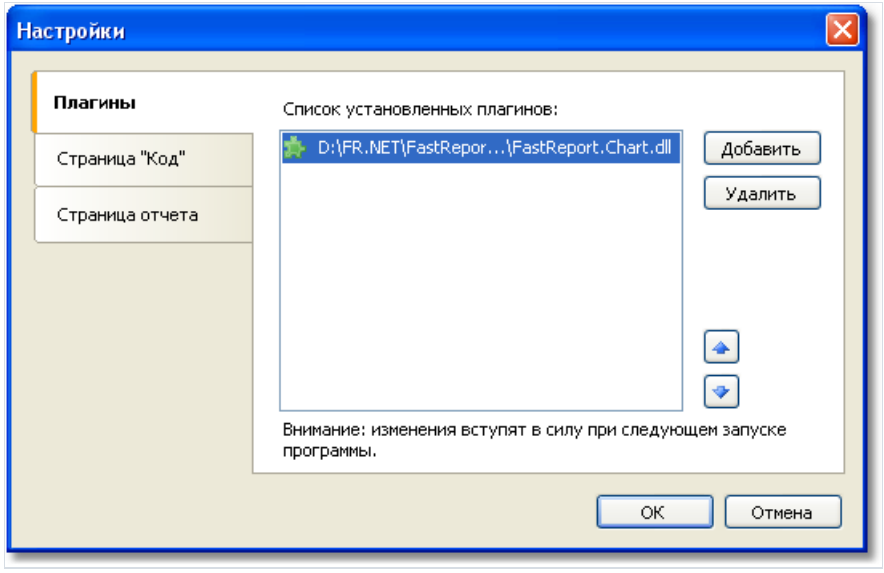
Этот код необходимо выполнить один раз за все время жизни приложения, перед запуском дизайнера.

Способ 2. Ваше подключение содержится в отдельном .dll модуле. В этом случае модуль можно подключить к FastReport, и он будет загружаться каждый раз, когда вы работаете с дизайнером. Чтобы это сделать, в модуле необходимо объявить публичный класс типа FastReport.Utils.AssemblyInializerBase, и в его конструкторе зарегистрировать свое подключение:

```
public class AssemblyInitializer : FastReport.Utils.AssemblyInitializerBase
{
  public AssemblyInitializer()
  {
    FastReport.Utils.RegisteredObjects.AddConnection(typeof(MyDataConnection));
  }
}
```

При загрузке вашего модуля FastReport найдет классы типа AssemblyInitializerBase и инициализирует их.

Чтобы подключить модуль к FastReport, вызовите дизайнер и в меню "Вид" выберите пункт "Настройки...". В открывшемся окне выберите закладку "Модули" и добавьте свой .dll модуль:



Это можно также сделать, добавив имя модуля в конфигурационный файл FastReport. Этот файл по умолчанию создается в каталоге пользователя:

```
C:\Documents and Settings\Имя_пользователя\Local Settings\Application Data\FastReport\FastReport.config
```

Добавьте модуль внутри тега Plugins:

```
<?xml version="1.0" encoding="utf-8"?>
<Config>
  ...
  <Plugins>
    <Plugin Name="c:\Program Files\MyProgram\MyPlugin.dll"/>
  </Plugins>
</Config>
```

Информация о выпусках

Версия 2024.1

Новые возможности

Улучшение работы с объектом "Таблица"

Работать с дизайнером отчетов стало проще и удобнее. Появились новые возможности для работы с объектом "Таблица".

1. Быстрое добавление столбцов и строк. Если подвести к границе между строками слева от таблицы, либо к границе между столбцами сверху, то появится условное отображение, показывающее, где будет добавлена новая строка или столбец. А также кнопка, по нажатию на которую новая строка или столбец будут добавлены в таблицу.

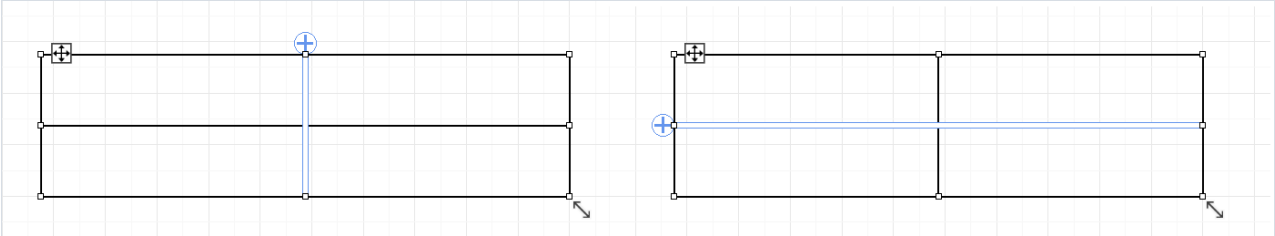
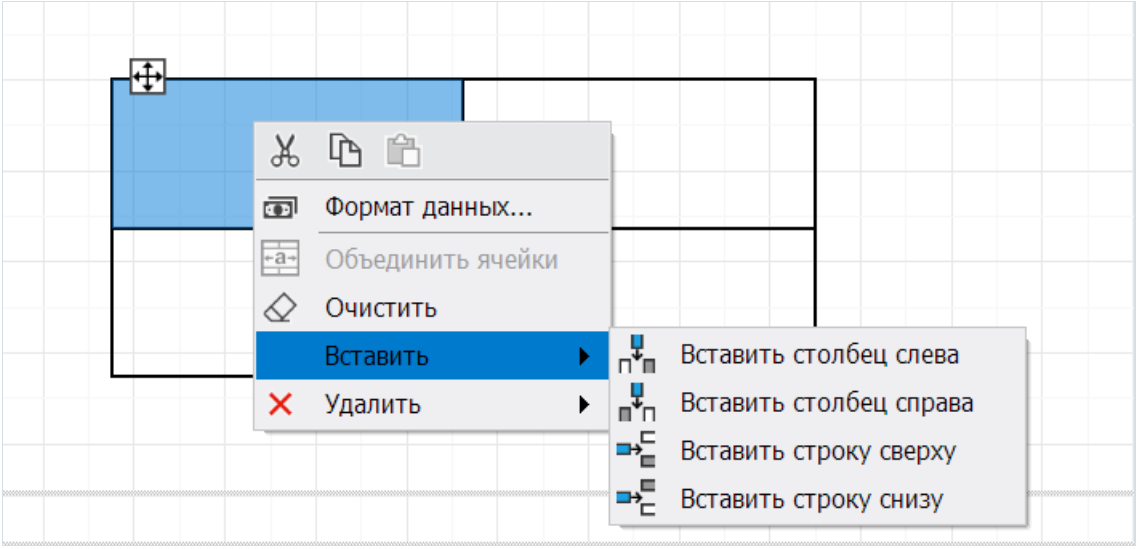


Таблица обязательно должна быть активна (выбрана), иначе новые элементы управления не появляются.

2. Изменение высоты строк и ширины колонок. Теперь перетаскивая границу столбца или строки с помощью мышки, можно изменять ширину столбца или высоту строки соответственно.

3. Выпадающее меню "Вставить". При нажатии правой кнопки мыши по ячейке в контекстном меню доступен выпадающий список, позволяющий вставить рядом с ячейкой новый столбец или строку.



4. Горячие клавиши. Копирование текста ячейки и вставка текста в ячейку с помощью горячих клавиш Ctrl + C и Ctrl + V**.**

Подробнее о новых возможностях таблиц читайте в статье.

Слияние текстовых объектов

Появился механизм слияния текстовых объектов с одинаковым текстом. Для этого у объекта "Текст" добавлено новое свойство MergeMode, которое позволяет настроить режим слияния. Работа нового свойства

очень похожа на свойство Duplicates в режиме Merge, но есть важные отличия:

- Duplicates работает только с одним объектом лежащим на бэнде "Данные". Например, на бэнде Data1 есть текстовый объект с именем Text1 и свойством Duplicates равным Merge. При построении отчета, на первой итерации Data1, при выводе первой записи в Text1 будет выведен текст "10". На второй итерации Data1 и выводе второй записи, в Text1 будет выведен такой же текст. В итоге два экземпляра Text1 будут соединены и текст "10" будет выведен только один раз.
- MergeMode, в отличие от Duplicates, может соединять экземпляры разных текстовых объектов. Причем делать это как по вертикали, так и по горизонтали. Например, при построении отчета текст "10" будет выведен в Text1 и расположенный рядом и справа от него Text2. При этом у Text1 свойство MergeMode равно Horizontal. В этом случае текстовые объекты будут соединены, и текст "10" будет выведен один раз.

Подробнее о новом свойстве читайте в статье по этой ссылке.

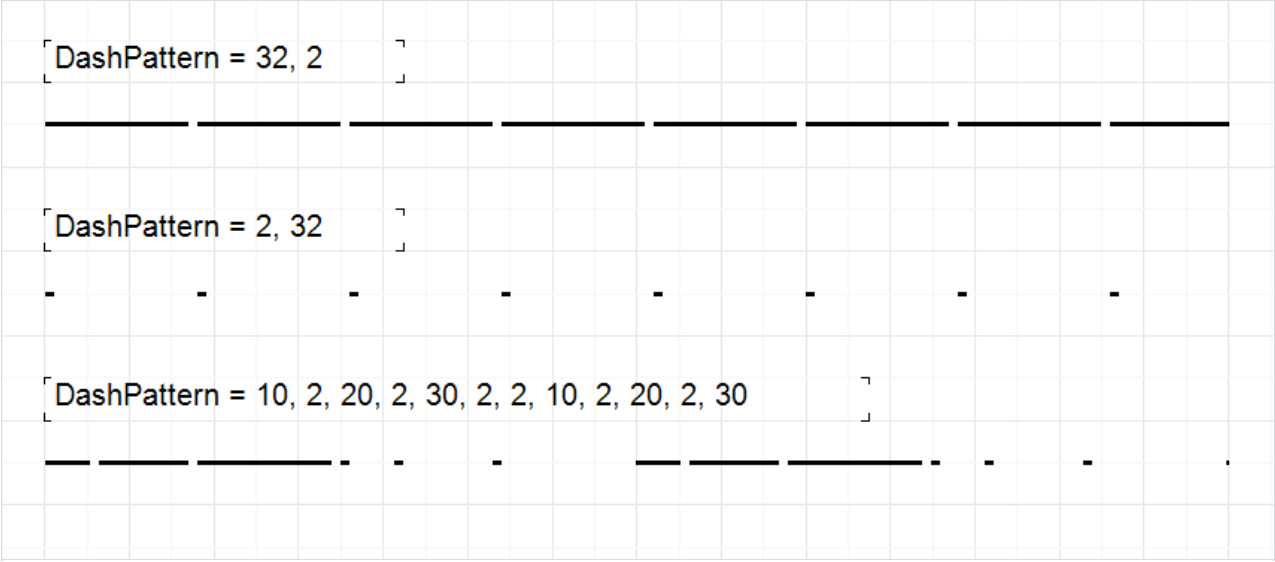
Пользовательские стили линий

Для объектов LineObject, ShapeObject, PolyLineObject и PolygonObject, добавлено новое свойство - DashPattern, которое позволяет создавать свои стили линий. Ранее стиль линий этих объектов настраивался с помощью свойства Border.LineStyle. Доступны были только шесть стилей: Solid, Dash, Dot, DashDot, DashDotDot и Double. С новым свойством можно указать коллекцию значений, которые будут последовательно задавать длину штрихов и пробелов.

Например, при значениях 5, 4, 3, 2 мы задаем паттерн при котором будет отображен штрих длиной 5, пробел длиной 4, штрих длиной 3 и пробел длиной 2. И далее значения будут повторяться по кругу, начиная с 5. Единица измерения здесь - Border.Width.

Если в коллекции DashPattern есть хотя бы одно значение, то будет работать этот новый механизм. А свойство Border.LineStyle будет игнорироваться. Если коллекция DashPattern пуста, то по прежнему будет работать механизм свойства Border.LineStyle.

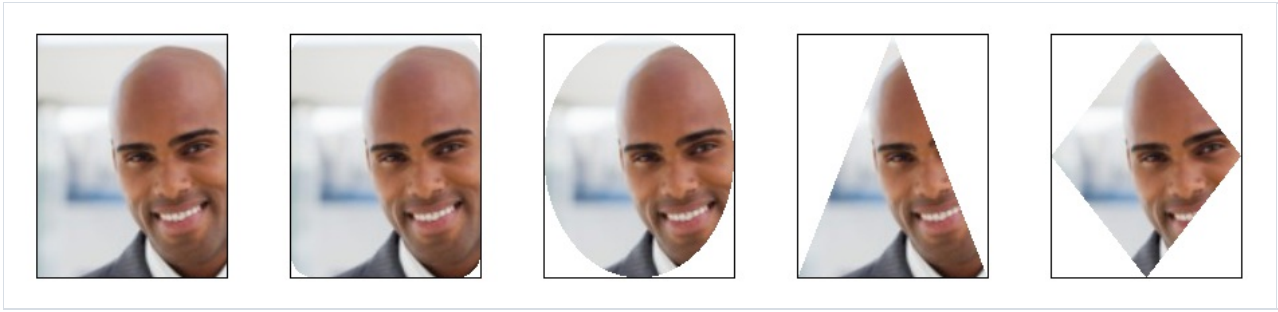
Ниже можно увидеть несколько примеров:



Инструкция по настройке линий доступна по следующей ссылке.

Изменение формы объекта "Рисунок"

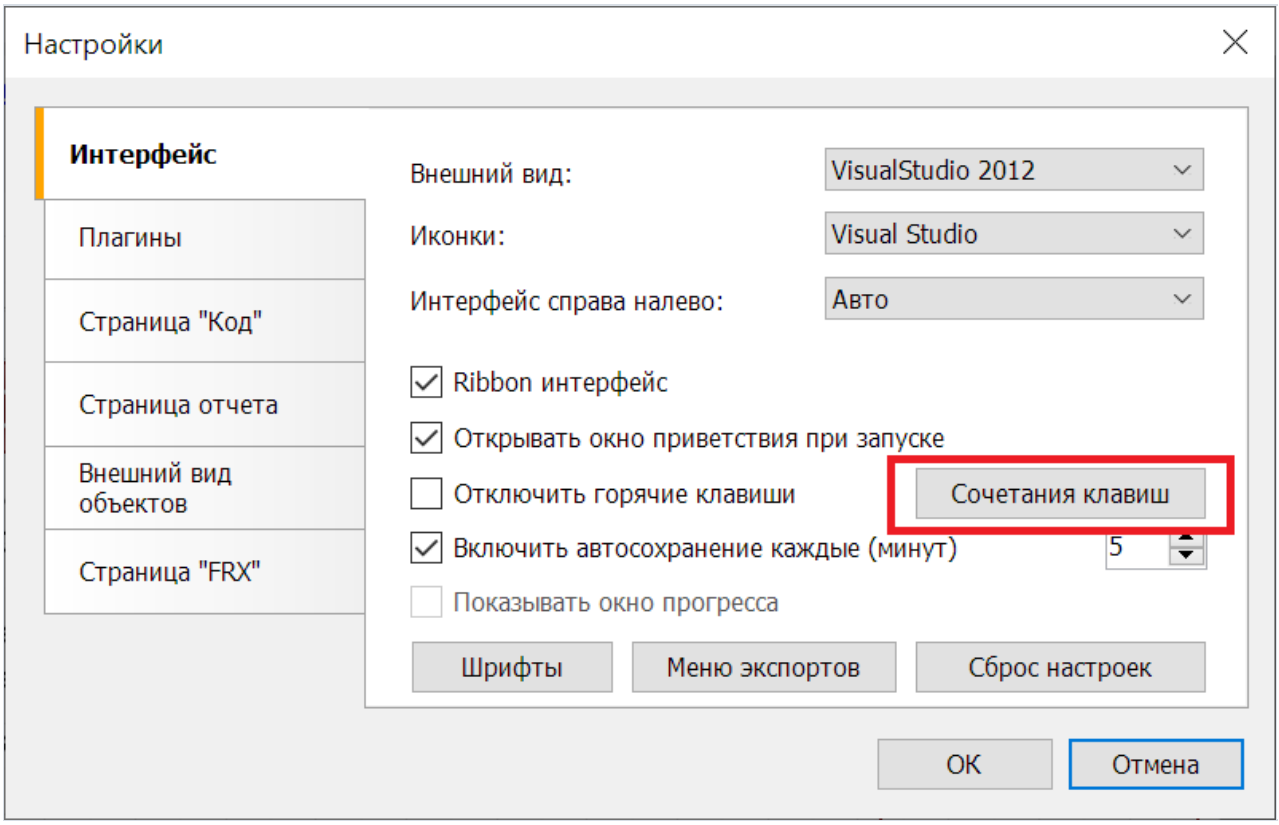
Появилась возможность изменять форму объекта "Рисунок". Теперь у PictureObject есть новое свойство Shape, которое позволяет задать следующие формы: прямоугольник (значение по умолчанию), прямоугольник со скругленными углами, эллипс, треугольник и ромб.



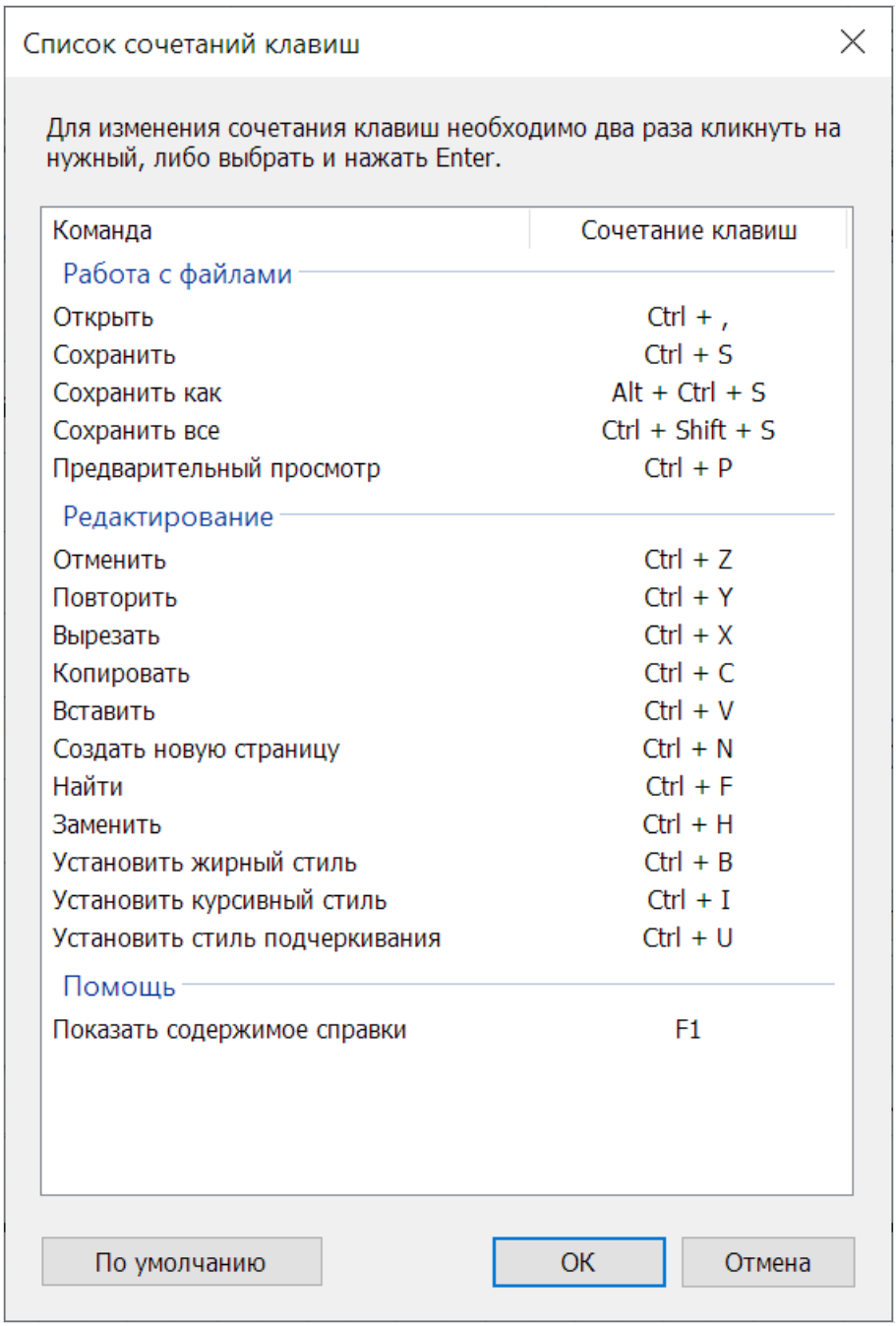
Подробнее можно узнать в статье.

Настройка комбинаций горячих клавиш

Появилась возможность настроить комбинации горячих клавиш по своему усмотрению. Можно настроить команды таких действий как "Открыть файл", "Сохранить файл", "Подготовить отчет" и многое другое. Для этого на вкладке "Интерфейс" в настройках дизайнера добавлена новая кнопка.



Нажатие которой, вызывает окно настройки комбинаций клавиш.



Здесь представлена таблица с действиями и назначенными им сочетаниям клавиш. Изменить комбинацию можно двойным кликом по нужной строке. В таблице также можно перемещаться с помощью клавиш "Вверх" и "Вниз", и вносить изменения нажатием клавиши "Enter". Также, можно вернуть все сочетания к значениям по умолчанию.

Больше информации о настройке клавиш вы можете найти в этой статье.

Поддержка .NET 8

Добавлена поддержка .NET 8 для FastReport .NET, FastReport.Core, FastReport.Core.Skia и FastReport.WPF. Эта платформа повышает производительность приложений и добавляет множество новых возможностей для ваших проектов.

Отказ от поддержки .NET Standard 2.0 в FastReport.Web

Для охвата всё большего количества технологий, которые постоянно добавляются в мир .NET, мы решили отказаться от устаревшего слоя совместимости .NET Standard 2.0 в нашей библиотеке для Web-интеграции FastReport.Web (WebReport Core/Skia). Теперь для этого продукта минимально поддерживаемая версия TargetFramework будет .NET Core 3.1 и выше (в том числе .NET 5, 6, 7 и 8). FastReport.Core и

FastReport.Core.Skia всё также без изменений будет поддерживать .NET Standard 2.0.

Добавлена поддержка ODBC коннектора для FastReport.Core

Наши пользователи долгое время просили нас добавить возможность подключения к базам данных через ODBC протокол для наших кроссплатформенных продуктов. Такая возможность присутствовала только в FastReport .NET и FastReport WPF ранее. С этим обновлением она доступна также в FastReport.Core и FastReport.Core.Skia. Для использования добавьте плагин FastReport.Data.Odbc в свой проект и зарегистрируйте его с помощью этого кода:

```
FastReport.Utils.RegisteredObjects.AddConnection(typeof(OdbcDataConnection));
```

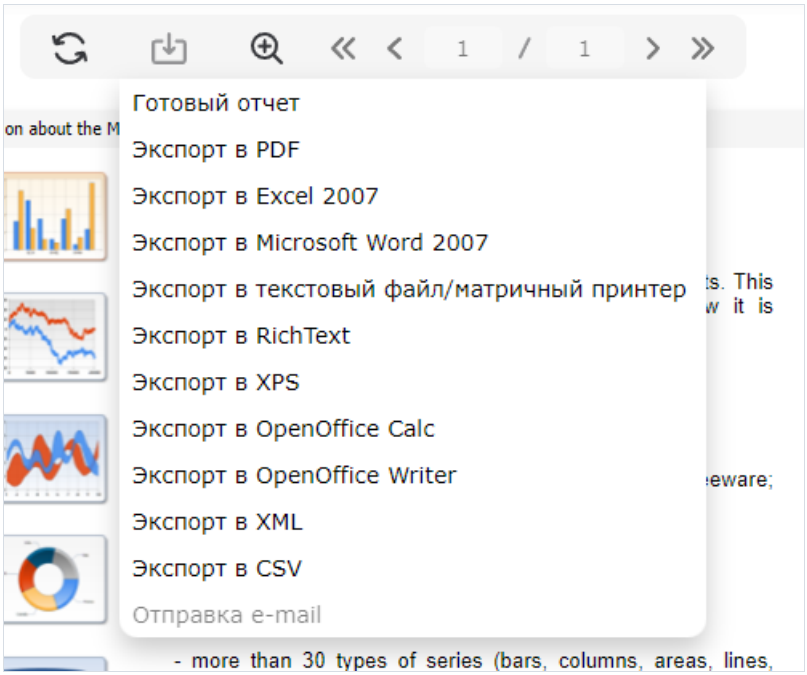
Изменения в WebReport

Email Экспорт в WebReport

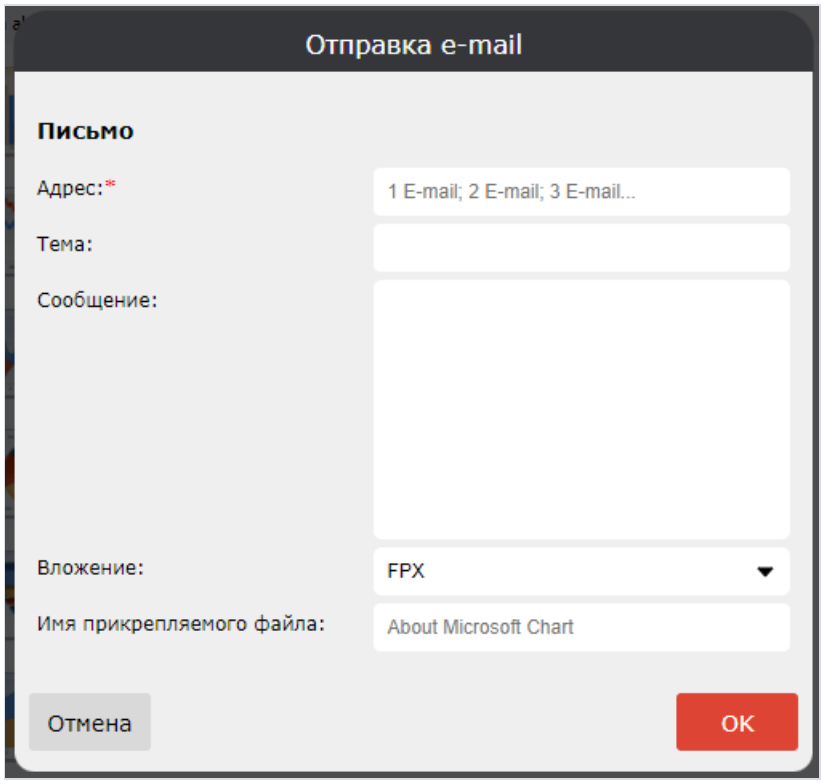
Теперь в WebReport появилась функция отправки отчётов по электронной почте. Для включения этой возможности необходимо настроить параметры SMTP-сервера при регистрации сервисов FastReport. Просто добавьте следующий код:

```
services.AddFastReport(options => options.EmailExportOptions = new      FastReport.Web.EmailExportOptions
            {
                Address = "SomeAddress@example.com",
                EnableSSL = true,
                Host = "Host",
                MessageTemplate = "Message template here",
                Name = "John",
                Password = "password",
                Port = 25,
                Username = "Username"
            });
```

После этого активируйте опцию WebReport.Toolbar.Exports.ShowEmailExport, и пользователи смогут отправлять отчёты по электронной почте:



При нажатии кнопки "Отправить по почте", пользователю будет предложено настроить сообщение через удобное модальное окно:

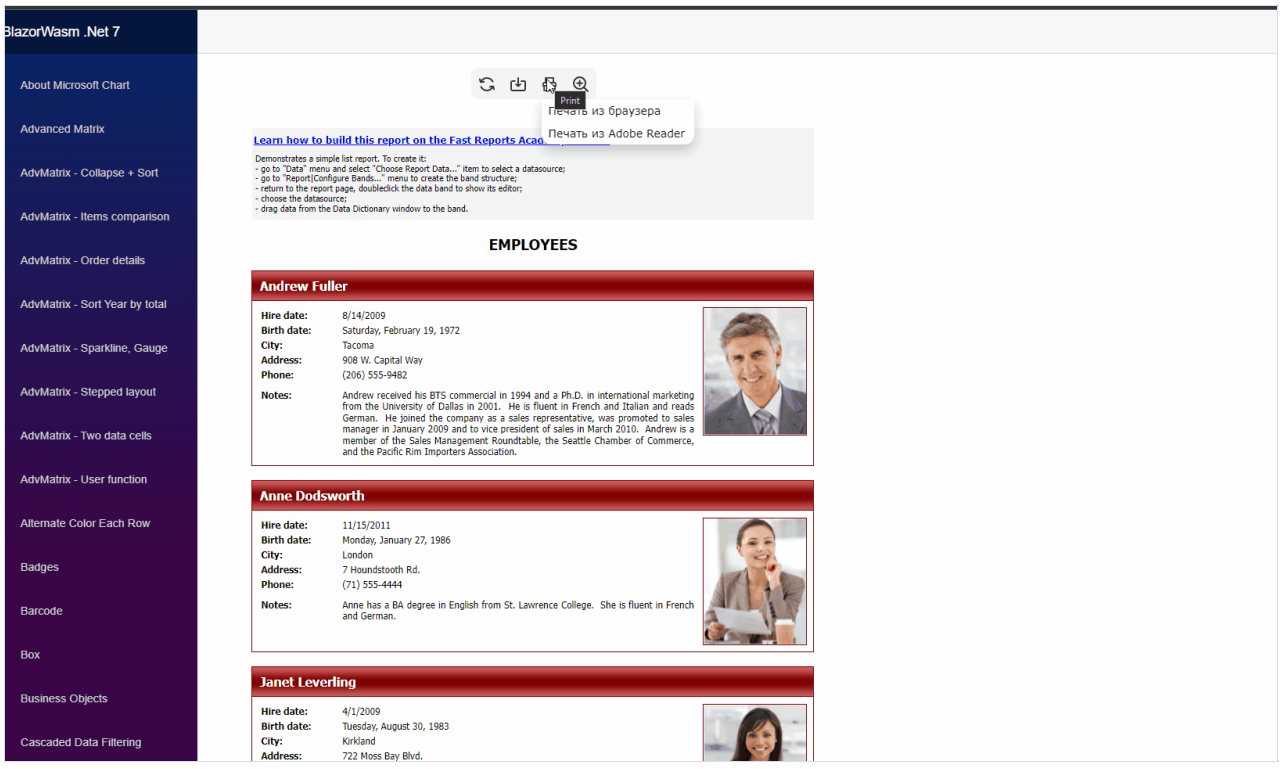


Печать в Blazor WebAssembly

Теперь WebReport позволяет печатать отчеты в Blazor WebAssembly. Эта функция по умолчанию включена, но если вам необходимо её отключить, просто используйте следующий код:

```
webReport.Toolbar.ShowPrint = false;
```

Теперь ваши отчеты могут быть напечатаны прямо из Blazor WebAssembly:



Полный список изменений

```
[Engine]
+ добавлено слияние текстовых объектов;
+ добавлена возможность изменять форму PictureObject;
+ добавлена возможность создавать пользовательские стили линий;
* теперь работа со шрифтами производится без блокировок;
- исправлен текст выходящий за границы TextObject при TextRenderer = HTMLParagraph;
- исправлено создание шрифтов из PrivateFontCollection;
- исправлен некорректный цвет текста в RichObject;
- исправлен разрыв RichObject с изображением;
- исправлена ошибка из-за которой пропадал фокус с объекта DateTimePicker, если у него было указано
свойство DetailedControl;
- исправлена ошибка в штрихкодах (отображение на HiDPI, экспорт в PDF);
- исправлен обратный отступ в HTMLTextRenderer;
- исправлены некорректные разрывы RichObject;
- исправлено обрезание текстового объекта при отрисовке с HTMLTextRenderer;

[Designer]
+ добавлена возможность убирать маркер у SberQR;
+ в настройки дизайнера добавлено свойство "Показывать окно прогресса";
+ добавлена возможность настраивать комбинации горячих клавиш;
+ повышено удобство работы с объектом "Таблица" в дизайнере;
* обновлены проверки для ссылок, теперь корректно обрабатываются ссылки с пробелами;
* изменено поведение деревьев - после очистки поля поиска дерево данных сворачивается, а дерево отчета
разворачивается;
- исправлено появление лишних линий при масштабировании RoundRectangle небольшого размера;
- исправлено кодирование косой черты в Barcode 93 Extended;
- исправлено удаление связи при слиянии словарей;
- исправлена ошибка с выбором форматирования даты или времени на венгерской локализации;
- исправлен сброс поиска в DataTree и ReportTree;
- исправлены исключения System.NullReferenceException и System.FormatException при вводе некорректных
данных в FloatCollection;

[Preview]
- исправлен некорректный размер границы страницы при бесконечной высоте или ширине страницы;

[Exports]
+ реализовано сохранение каждой картинки в отдельный поток;
+ добавлены недостающие ссылки на обработчики событий в экспортах в Excel 2007, Word 2007 и RTF;
+ добавлено новое свойство для масштабирования штрих-кодов при экспорте в ZPL;
+ добавлен выбор группы, по которой отчет будет разделен на листы в Excel 2007;
+ добавлена возможность отключать группировку листов при экспорте в Excel 2007;
+ добавлено использование режима переноса для текстурной заливки при экспорте в SVG;
* при экспорте в облачные хранилища окно автоматически закрывается после получения кода авторизации;
- исправление приватных коллекций шрифтов;
- исправлена ошибка разбора таблицы GSUB;
- исправлен некорректный экспорт стилей границ объектов DashDot, DashDotDot и Double в PDF;
- исправлена ошибка, из-за которой числа в Gauge при HTML экспорте отображались размыто;
- исправлено вычисление заголовка ContentMD5 в S3 экспорте;
- исправлено некорректное позиционирование текста при экспорте в ZPL;
- исправлен некорректный экспорт GaugeObject в PowerPoint 2007;
- исправлен некорректный экспорт RadialGauge с заливкой в послойный экспорт в Word 2007;
- исправлен некорректный экспорт RadialGauge с заливкой в не послойный HTML;
- исправлено отображение в режиме переноса "Clamp" для текстурной заливки при экспорте в SVG;
- исправлено изменение размера текста при использовании HTML тегов в Excel 2007 экспорте;
- исправлено некорректное поведение HTML тегов с табуляцией при экспорте в Excel 2007;
- исправлена проблема снижения качества водяного знака при экспорте в PDF;
- исправлена ошибка с некорректными отступами при табличном экспорте в Word 2007;
- исправлено позиционирование изображения в CheckBox при экспорте в Word 2007;

[WebReport]
+ добавлена возможность отправить письмо с отчётом из WebReport;
+ добавлена возможность печати в FastReport.Blazor.WASM;
- убрана поддержка .NET Standard 2.0 в FastReport.Web;
- исправлен баг из-за которого появлялась ошибка при экспорте в приложении Blazor;
- исправлено игнорирование Margin при печати с PrintHtml в WebReport;
```

```
[.Net Core]
- исправлена ошибка, из-за которой при экспорте в PDF неверно рассчитывалась ширина текста;

[Demos]
- исправлена ошибка отображения навигационного меню после сворачивания Demo New;

[Extras]
+ добавлено преобразование Variant в типы CLR в MySqlDataConnection;
+ добавлен плагин FastReport.Data.Odbc;
+ добавлена поддержка FastReport.WPF для плагинов-коннекторов FastReport.Data.*;
* изменено поведение сообщения о повторяющихся именах в запросе;
- исправлена работа автоматического создания параметров в запросе.
```

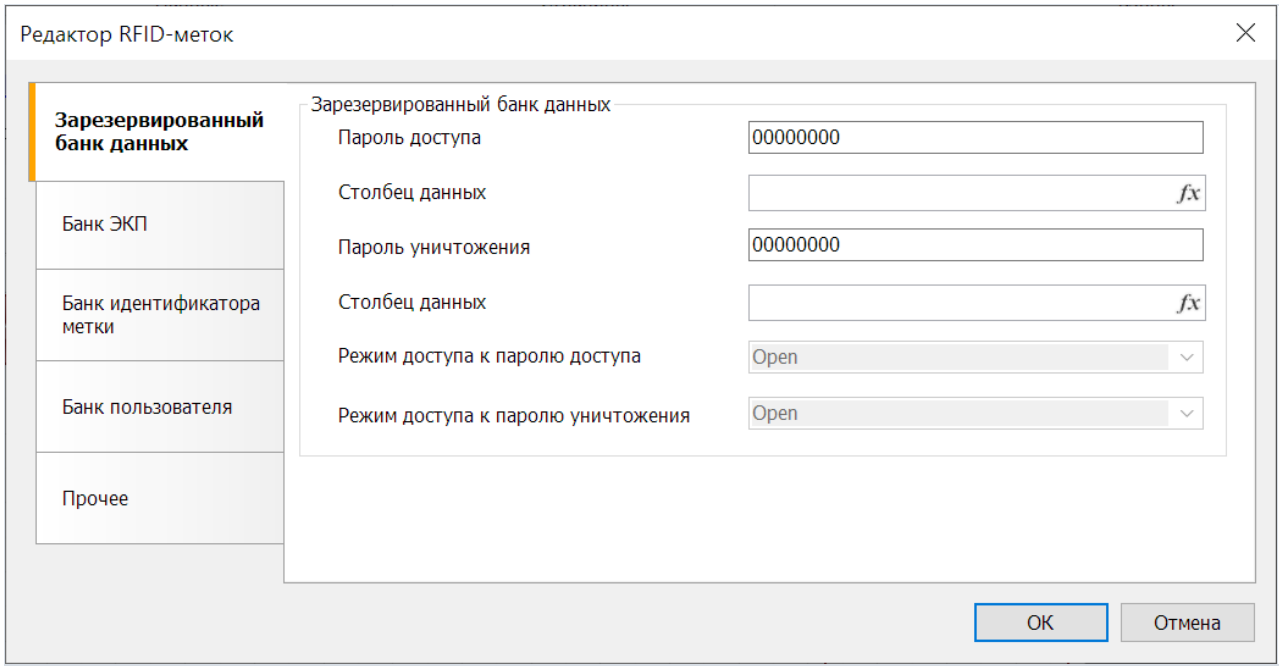
Версия 2023.3

Новые возможности

Новый объект RFIDLabel

В новой версии появился объект - RFID-метка. Она позволяет идентифицировать товары и очень похожа на штрих-код, но в отличии от него использует радиосигналы. Это позволяет сканировать большое количество товаров за короткие промежутки времени.

Метка содержит 4 банка данных: зарезервированный банк для хранения пароля доступа и пароля уничтожения, банк электронного кода продукта, банк идентификатора метки и банк пользовательских данных. Метка в линейке продуктов FastReport .NET, представлена в виде объекта отчета. Настраивать метку можно с помощью удобного редактора, который вызывается двойным щелчком мыши.



RFID-метки могут быть созданы некоторыми принтерами компании Zebra, поэтому кроме самого объекта меток, был реализован их экспорт в ZPL. Для корректного экспорта RFID-метка должна быть в единственном экземпляре на странице.

Подробнее читайте в статье.

Поддержка изображений в формате WebP

Появился плагин, обеспечивающий поддержку изображений в формате WebP. Теперь можно загружать такие изображения в объект PictureObject с помощью редактора в дизайнере отчетов, а также из кода. В FastReport.Skia WebP-изображения поддерживаются без плагина, однако они при загрузке преобразовываются в формат PNG.

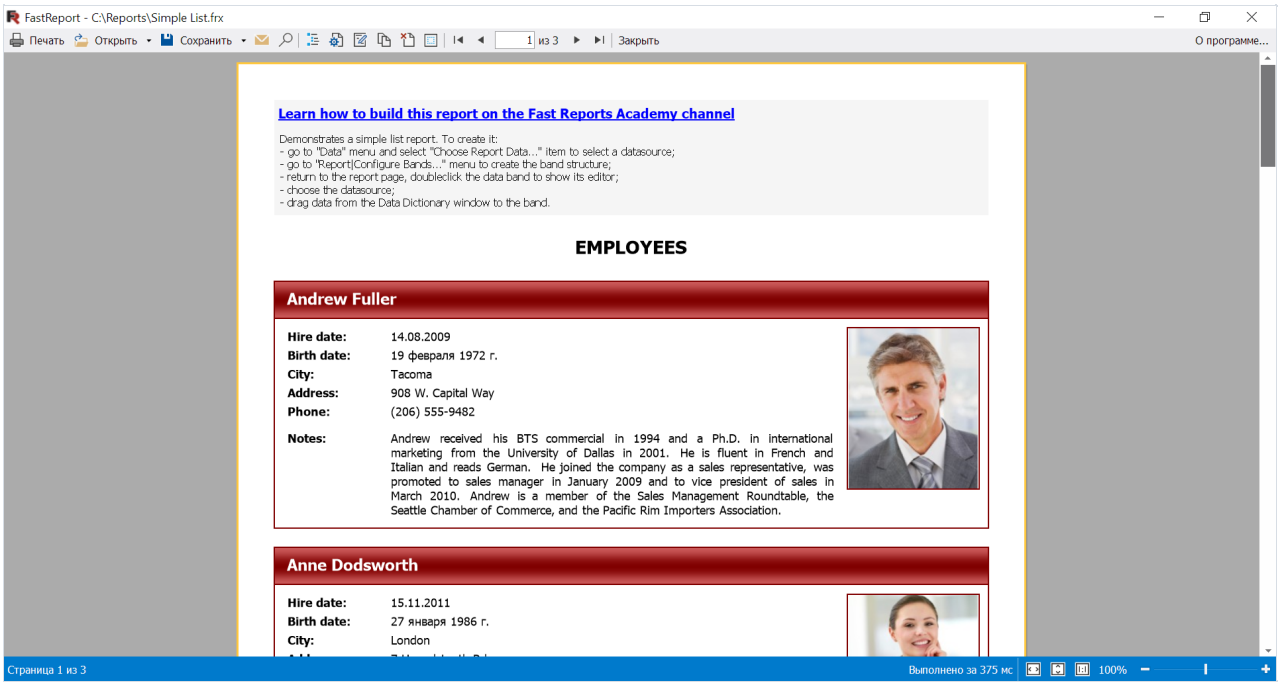
Подробности о формате и инструкцию по использованию плагина, можно найти в статье.

Предварительный просмотр внутри окна дизайнера и асинхронный просмотр отчета

Теперь есть возможность, при использовании дизайнера в своем приложении, запускать предварительный просмотр отчета внутри окна дизайнера. Ранее предварительный просмотр всегда запускался в отдельном окне. Для этого нужно в коде добавить строку:

```
Config.DesignerSettings.EmbeddedPreview = true;
```

Выглядеть это будет так:



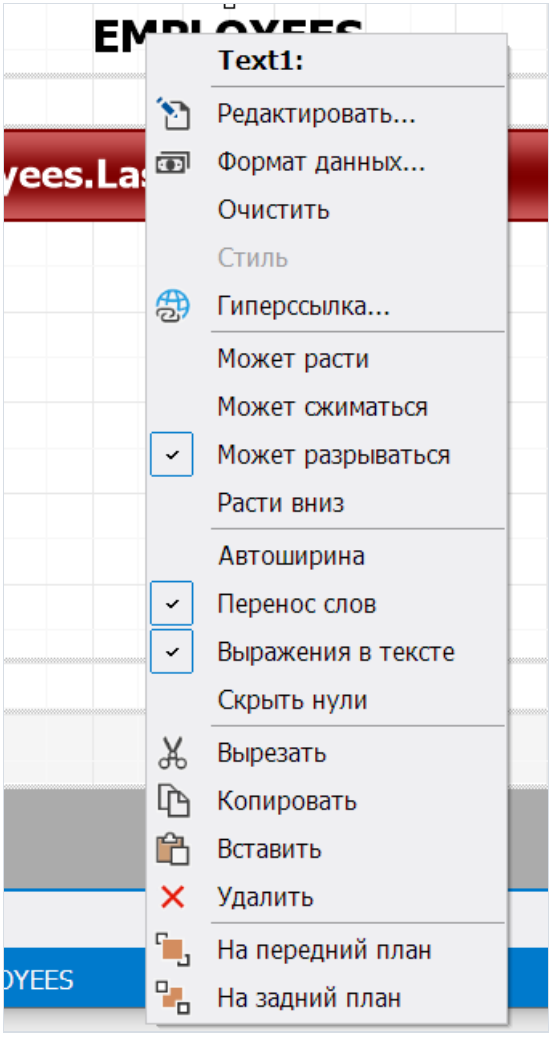
В некоторых случаях, такой режим может быть удобнее.

Кроме этого, появились методы асинхронной подготовки и просмотра отчета: Report.PrepareAsync() и Report.ShowAsync(). Их можно использовать при подготовке больших отчетов. В таком случае, с окном предварительного просмотра можно будет работать, пока идет подготовка отчета. И у пользователя не будет создаваться впечатление, что приложение зависло и не отвечает.

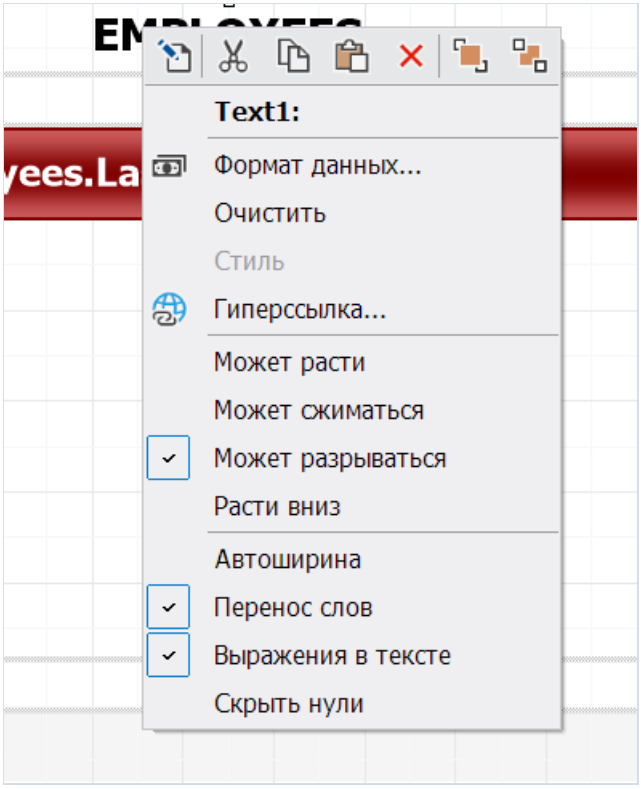
Панель инструментов в контекстном меню

Было улучшено контекстное меню при нажатии правой кнопки мыши по объекту. Вверху появилась панель инструментов, в которую вынесены часто используемые пункты, такие как: редактировать, вырезать, копировать, вставить и т.д.

Раньше меню выглядело так:



Новое меню стало компактнее и эргономичнее:

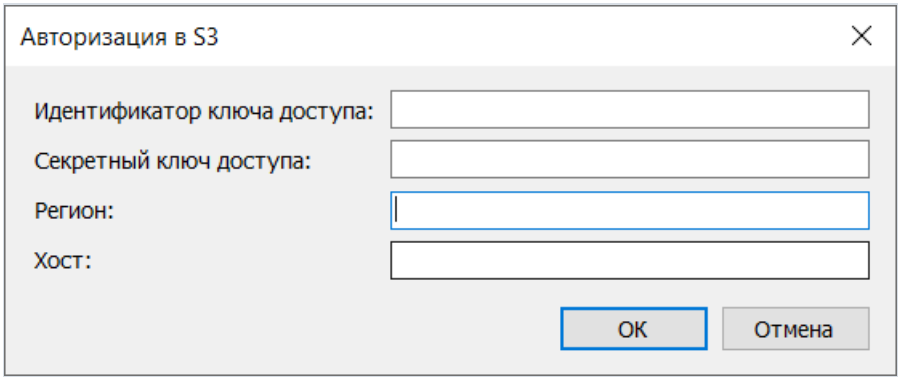


Экспорт в S3

Добавлена возможность выгружать подготовленные и экспортированные отчеты в Simple Storage Service (сокращенно S3). Новый экспорт находится во вкладке "Хранилище" меню сохранения подготовленного

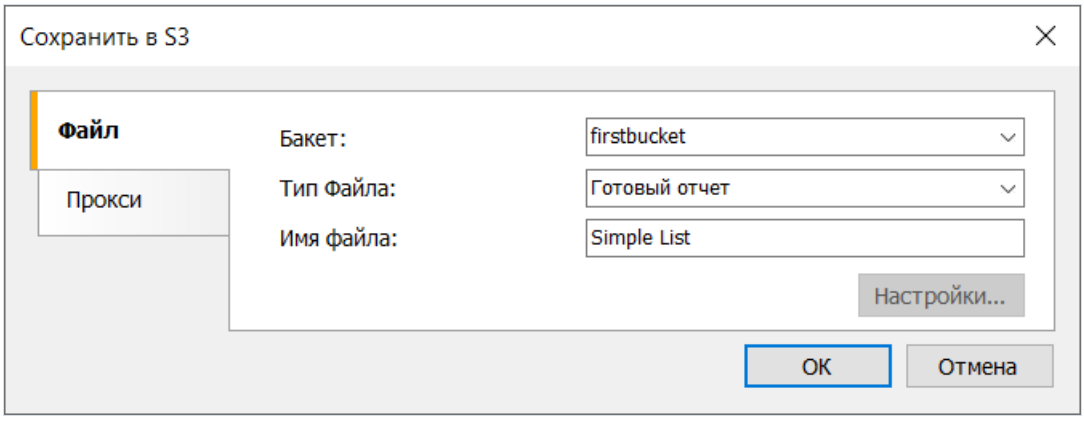
отчета.

При первом экспорте понадобится ввести регистрационные данные в окне авторизации.



Необходимые ключи можно получить в настройках аккаунта сервиса S3. Подробнее можно почитать в документации сервиса.

После успешной авторизации появится окно экспорта.



Здесь можно выбрать бакет для сохранения, тип и имя файла. Если выбрать тип файла отличный от "Готовый отчет", то станут доступны настройки соответствующего экспорта.

Подробнее читайте в статье.

Возможность настраивать параметры шрифта штрихкодов

Для объектов "Штрих-код" теперь доступно свойство "Font". Оно позволяет задать параметры шрифта, используемого при отображении текстов штрих-кодов. По умолчанию используется шрифт Arial, он же использовался в прошлых версиях. Теперь можно выбрать другой шрифт, изменить его размер, начертание и т.д. В результате можно создавать, например такие штрих-коды:

Однако, с настройками шрифта следует быть осторожным. Возможно, не все сканеры смогут корректно считывать такие штрих-коды.

Опция "Преобразовывать общий формат в текстовый" при экспорте в Excel 2007

В Excel 2007 есть несколько форматов данных, среди которых есть два очень похожих: общий и текстовый.

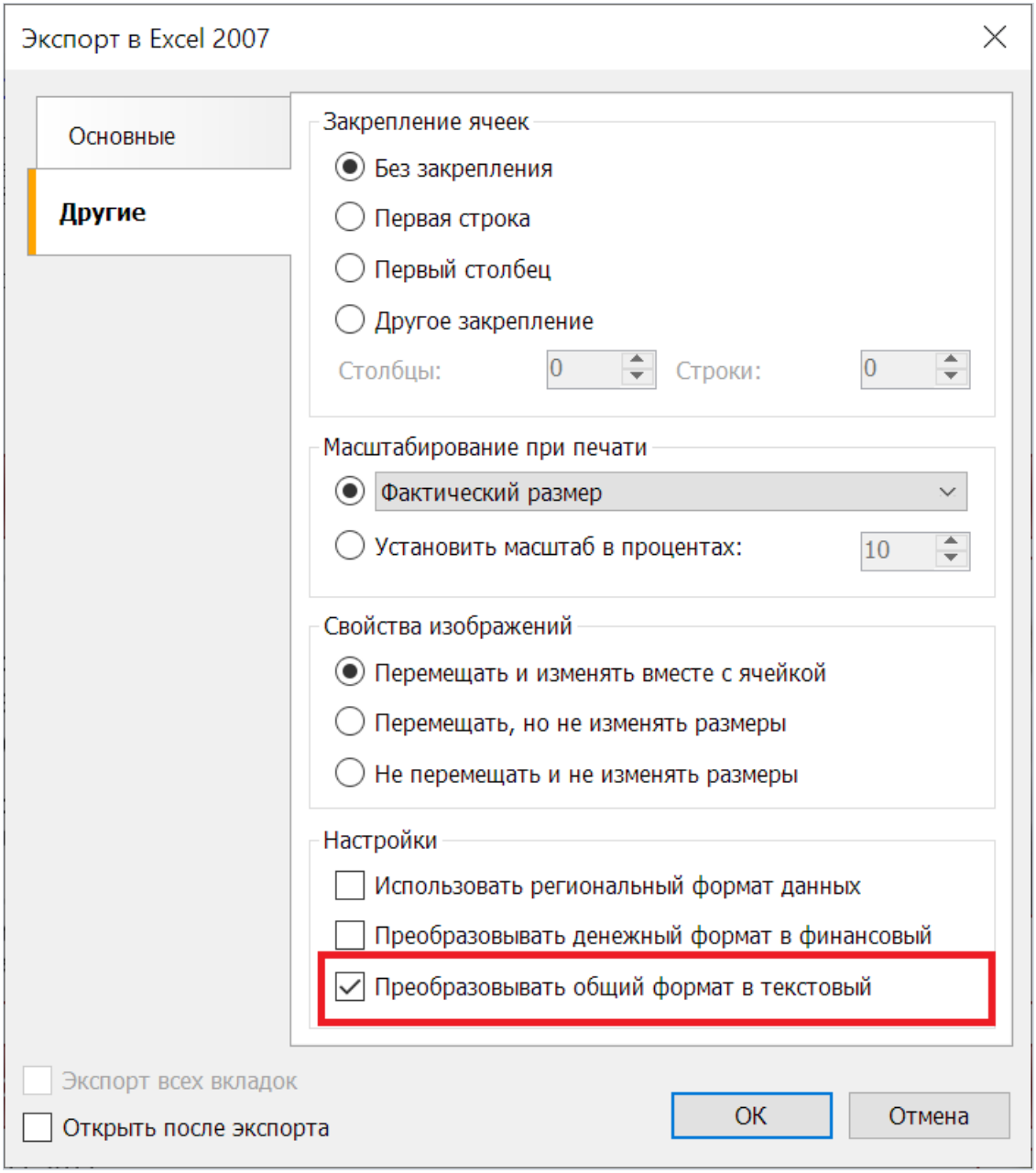
Общий используется по умолчанию. В большинстве случаев числа в этом формате отображаются так, как вводятся. Но если ширины ячейки недостаточно для отображения всего числа, то оно округляется.

Текстовый формат всегда отображает данные так, как они введены.

В FastReport .NET тоже есть несколько форматов, например: общий, числовой, дата и многие другие. При экспорте для данных подбирается подходящий формат, числовой преобразовывается в числовой, дата остается датой.

Общий формат в FastReport .NET тоже используется по умолчанию. Он отображает данные так же, как они были введены. На самом деле общий формат это System.String. В свою очередь отдельного текстового формата в FastReport .NET не существует.

В Excel 2007-экспорте появилась новая опция, которая позволяет преобразовывать общий формат FastReport .NET в текстовый формат Excel (по умолчанию общий экспортируется как общий).



Отчеты, созданные в прошлых версиях FastReport .NET, в новой версии будут экспортироваться так же, так как по умолчанию эта опция отключена.

Поддержка частичной компиляции отчёта

В FastReport.Core, FastReport.Core.Skia и FastReport.CoreWin была добавлена возможность включать частичную компиляцию отчёта для ускорения его подготовки в случае, если в отчёте не изменялся скрипт отчёта и нет объектов, которые не поддерживают частичную компиляцию. Включить его можно следующей командой:

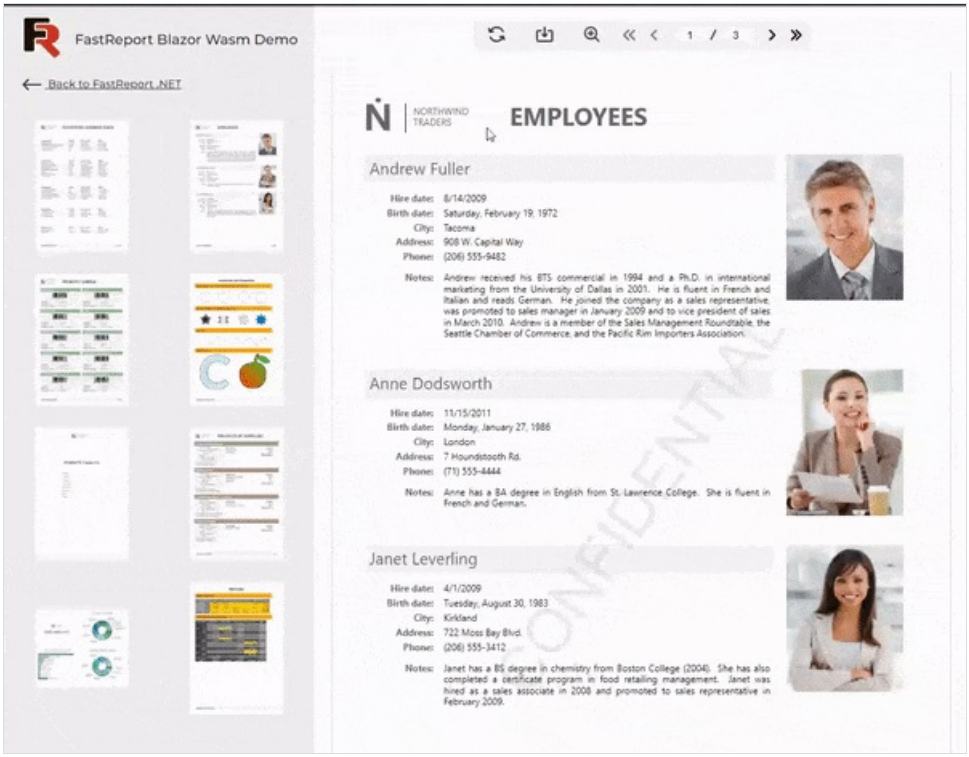
```
FastReport.Utils.Config.CompilerSettings.ReflectionEmitCompiler = true;
```

Включение Reflection.Emit компилятора не вызывает ухудшений в работе. В случае, если новый компилятор не может быть использован в новом отчёте, то он просто будет использовать стандартную процедуру без вреда для отчёта.

Новые возможности WebReport

Улучшения в WASM

Ранее отчёты, открытые в браузере при помощи нашей библиотеки FastReport.Blazor.Wasm, можно было только просматривать. С этим обновлением была добавлена поддержка экспортов. Теперь пользователи могут экспортировать получившиеся отчеты в различные форматы, точно также, как это делается и в обычным WebReport.



Также, благодаря Reflection.Emit компиляции, была ускорена загрузка и подготовка отчётов без скрипта в WebAssembly.

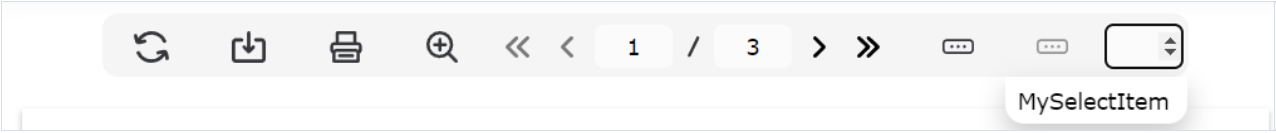
Персонализация панели инструментов

Теперь вы можете создавать свои собственные элементы, такие как кнопки, выпадающие списки и поля для ввода, и добавлять их в панель инструментов. Эти элементы могут иметь различные параметры, включая изображение, название и стили. Вы также можете внедрять логику с использованием JavaScript и C#.

Пример добавления собственных элементов в панель инструментов:

```
var button = new ToolbarButton()
{
    Title = "MyCustomButton",
    OnClickAction = new ElementClickAction()
    {
        OnClickAction = async (webreport) =>
        {
            webreport.LocalizationFile = "MyLocalizationFile";
        }
    },
};

var select = new ToolbarSelect()
{
    Title = "MyCustomSelect",
    Items = new List<ToolbarSelectItem>
    {
        new ToolbarSelectItem()
        {
            Title = "MySelectItem",
            OnClickAction = new ElementScript()
            {
                Script = "console.log('My item is working')"
            }
        }
    }
};

var input = new ToolbarInput()
{
    InputType = "number",
    OnChangeAction = new ElementChangeAction()
    {
        OnChangeAction = async (webreport, inputValue) =>
        {
            webreport.Report.Prepare();
            webReport.Toolbar.Height = int.Parse(inputValue);
        }
    }

};

webReport.Toolbar.InsertToolbarElement(button);
webReport.Toolbar.InsertToolbarElement(select);
webReport.Toolbar.InsertToolbarElement(input);
```

Как результат, после внесения изменений, эти настроенные элементы будут отображаться в вашей панели инструментов:



Полный список изменений

```
[Engine]
+ добавлен новый объект RFIDLabel;
+ добавлено автоматическое форматирование GS1 для штрих-кода GS1-128;
+ добавлена загрузка таблиц в ячейках других таблиц при конвертации шаблонов RDL;
+ добавлено свойство Config.CompilerSettings.ReflectionEmitCompiler, которое при включении ускоряет
подготовку отчета если не был изменен скрипт (работает только в .NET Core/.NET);
+ добавлена возможность настраивать параметры шрифта штрих-кодов с помощью нового свойства "Font";
```

+ добавлена возможность настраивать параметры шрифта штрих-кодов с помощью нового свойства "Font";
* улучшена работа с приватными коллекциями шрифтов;
* демо версия - убрано ограничение в 5 страниц; текст заменяется случайным образом на "Demo version";
- исправлен бесконечный цикл при вычислении выражения параметра, равного этому параметру;
- исправлена проблема считывания штрих-кода DataMatrix мобильным сканером;
- исправлен баг из-за которого, при ручных переносах некорректно отображались зачеркивания строк;
- исправлено вычисление сдвига транслируемых объектов RichObject;
- исправлено преобразование пустого Variant в другие типы;
- исправлено удаление столбца, после которого данные столбца оставались в отчете;
- исправлена работа свойства VisibleExpression для строк и столбцов матрицы и таблицы;
- исправлено удаление больше не присутствующих шрифтов из словаря font_hash;
- исправлена ошибка при неотсортированных позициях табуляции в RichObject;
- исправлена ошибка с разбором таблицы GSUB, приводящая к исключению;
- исправлена потеря позиции потока при экспорте в PDF с опцией "Текст в кривых", приводящая к System.StackOverflowException;
- исправлена ошибка загрузки границ объектов при конвертации шаблонов RDL;
- исправлено удаление первых трех символов в штрих-коде GS1-128;
- исправлена таблица кодирования штрих-кода Code93 Extended;
- исправлена кодировка текста в штрих-коде DataMatrix;
- исправлена ошибка в отрисовке текста при переносе слов из-за недостатка места;
- исправлено преобразование RightToLeft текста при включенной опции ConvertRichText;
- исправлен перенос строки в HtmlTextRenderer;
- исправлена ошибка, когда столбцы страницы печатались поверх столбцов бэнда;
- исправлено выделение белым цветом пустых строк между абзацами текста и некоторых абзацев в RichObject при использовании заливки;
- исправлено выделение частей текста белым цветом в RichObject с ConvertRichText = true;
- исправлено игнорирование ConnectionString, если ConnectionStringExpression возвращало значение null;
- исправлены отступы транслируемых текстовых объектов из RichObject;
- исправлено позиционирование объектов при транслировании RichObject;
- исправлен импорт таблиц из JasperReports;
- исправлено исключение System.NullReferenceException при очистке TableObject;
- исправлено горизонтальное выравнивание картинки в RichObject при ConvertRichText = true;
- исправлено исключение System.NotImplementedException, когда позиция табуляции TextObject отрицательна;
- исправлено преобразование нулей, если выражение содержит функцию;
- исправлено исключение System.ArgumentException, когда хост источника данных JSON имеет пустой CharacterSet;
- исправлено позиционирование TableObject при транслировании RichObject;

[Designer]
+ добавлена возможность взять имена столбцов из первой строки в Excel-подключении;
+ добавлены категории для объектов "Штрих-код";
+ добавлено свойство Config.DesignerSettings.EmbeddedPreview для предварительного просмотра отчета внутри окна дизайнера;
+ в панели "Объекты" добавлена категория "Другие" для диалоговых элементов управления;
+ добавлена возможность отображать транслированный объект в Онлайн Дизайнере;
+ добавлена страница выбора процедур в форме мастера подключения к данным;
+ добавлена панель инструментов в контекстное меню;
+ добавлена возможность использовать выражения в поле "Сумма платежа" в редакторе SberbankQr;
+ добавлен парсинг параметров из SQL-запроса;
+ добавлено предупреждение при совпадении имен параметров запроса;
+ добавлена проверка на существование файла при его изменении в CSV-подключении через свойство CsvFile;
* изменения в интерфейсе "Построителя запросов";
* обновлен "Мастер подключения к данным". Улучшен интерфейс, исправлены ошибки и увеличена скорость работы;
* изменение отрисовки подсказки с координатами/размерами в дизайнере;
- исправлена проблема подключения к CSV через URL;
- исправлена ошибка в операции "Сохранить как..." для файла, открытого из облака;
- исправлен объект "Карта" в NET 6.0 (пустые надписи у полигонов);
- исправлена ошибка считывания значений из файла конфигурации дизайнера;
- исправлен баг при котором создавалась новая страница отчета после двойного клика правой кнопкой на вкладке "Код";
- исправлена ошибка после закрытия окна предварительного просмотра, при пустых значениях числовых параметров;
- исправлен баг, при котором в процессе авторизации зависал дизайнер;
- исправлены ошибки в редакторах объектов Gauge;
- исправлено исключение System.NullReferenceException при объединении словарей, которые включают соединения с параметрами;
- исправлено выделение текста цветом у RichObject при использовании свойства ConvertRichText = true;
- исправлена ошибка с порядком форматов при нескольких выражениях в текстовом объекте;

- исправлена ошибка масштабирования в окне настроек дизайнера на вкладке "Плагины";
- исправлено некорректное масштабирование формы выбора источника данных в Visual Studio;
- исправлено неполное отображение страниц с бесконечной шириной в предварительном просмотре добавления страниц;
- исправлен баг загрузки отчета с паролем;
- исправлены проблемы с масштабированием некоторых контролов;
- исправлена ошибка когда выбраны поля у невыбранных таблиц при редактировании подключения;
- исправлена ошибка когда выбраны все таблицы при редактировании подключения, хотя на самом деле выбраны только некоторые из них;
- исправлено исключение System.IO.FileFormatException при использовании неправильного XML отчета на странице FRX;
- исправлена некорректная работа настроек шрифта в MSChartObject при масштабировании больше 100%;
- исправлена ошибка при подключении базы данных CSV через URI;
- исправлена ошибка при запуске отчета с MSChartObject и SparklineObject на DataBand с включенным свойством CanBreak;
- исправлены проблемы с отображением SVG в дизайнере;
- исправлена ошибка с размером шрифта в окне "Дерево отчета";
- исправлено поведение окна "О программе" при изменении масштабирования;
- исправлено игнорирование отрисовывания MSChartObject при отсутствии Title;

[Preview]
- исправлено выравнивание текстового объекта по горизонтали при AutoWidth = true;
- исправлены проблемы с отображением SVG в предварительном просмотре;

[Exports]
+ добавлен экспорт в S3;
+ добавлен экспорт границ страницы при Image-экспорте;
+ добавлена опция "Использовать разрывы страниц" в форме экспорта в HTML;
+ добавлена опция позволяющая включать и отключать добавление закладок на каждую страницу при экспорте в Word 2007;
+ добавлено создание нового листа при приближении количества строк к максимально допустимому на одном листе в Excel 2007;
+ добавлена опция "Преобразовывать общий формат в текстовый" в Excel 2007 экспорте;
+ расширение имен шрифтов;
+ улучшена подсистема упаковки шрифтов для экспорта в PDF;
* ускорена работа экспорта в PDF;
* оптимизирован экспорт интерактивных форм в PDF;
- исправлена ошибка, из-за которой не учитывался LineHeight при экспорте с использованием Skia;
- исправлен многопоточный экспорт в PDF и частные коллекции шрифтов;
- исправлена загрузка шрифтов с традиционными китайскими иероглифами;
- исправлен кернинг шрифтов с написанием справа налево при экспорте в PDF;
- исправлена ошибка при которой шрифты, размером меньше 10, отображались некорректно с включенным свойством ConvertRichText при экспорте в RTF;
- исправлены ошибки кернинга в PDF экспорте;
- исправлена ошибка в PDF экспорте в режиме "Текст в кривых" на высоком разрешении монитора;
- исправлена ошибка, когда для некоторых объектов в PDF экспорте рисовалась тёмная рамка;
- исправлен экспорт семейств шрифтов, зарегистрированных в FastReport.Utils.FRPrivateFontCollection;
- исправлено отображение HTML тегов <strike>, <sub> и <sup> при экспорте в RTF;
- исправлен баг, при котором экспорт отчета с картинками под Skia завершался ошибкой;
- исправлен экспорт объектов подвала в RTF и DOCX;
- теперь однобайтовые пробелы не пропадают из строки после экспорта в Excel 2007;
- исправлено добавление лишних переносов текста при экспорте в CSV;
- исправлена ошибка с добавлением лишних разделителей при экспорте в CSV;
- исправлена ошибка, из-за которой повреждались шрифты при многопоточном экспорте в PDF;
- исправлена ошибка из-за которой не обрабатывались символы переноса при экспорте в HTML;
- исправлена некорректная работа гиперссылок в RichObject при экспорте в PDF;
- исправлен множитель высоты строки в RTF экспорте;
- исправлено двойное сохранение отчета в Google Drive;
- исправлен вызов API для сохранения отчетов в OneDrive;
- исправлены проблемы отображения SVG при экспорте в PDF;
- исправлены ошибки в дереве экспортов;
- исправлен экспорт текста с HTML-тегами в Word 2007;

[WebReport]
+ добавлена тень для отчёта в WebReport;
+ добавлена поддержка экспорта отчётов в Wasm;
* изменено поведение Toolbar для отчётов с одной страницей;
* изменено поведение печати отчёта из браузера в WebReport. Теперь страница печати закрывается автоматически;

```
- исправлена ошибка из-за которой не работало события клика в WebReport;
- исправлен некорректный экспорт в Word 2007 в веб отчетах;
- исправлена ошибка, при которой некоторые объекты отчета (например, RichObject) могли не отображаться в
Web-дизайнере;
- исправлена ошибка, при которой не происходил экспорт одностраничного отчёта, если использовались
настройки;
- исправлена ошибка, из-за которой не обновлялся отчёт при изменении параметра;

[.NET Core]
- исправлена ошибка при включении опции InvariantGlobalization;

[Demos]
* изменен скрипт в шаблоне "Sort Group By Total" для корректной работы отчета и отображения итогов, при
использовании свойств "Может расти", "Может сжиматься" бэнда "Подвал группы";

[Extras]
+ добавлен экспорт границ страницы при экспорте с помощью PDFSimpleExport;
+ добавлена возможность подключения к MariaDB с помощью плагина MySqlConnection;
+ добавлен формат .db в файловый фильтр для подключения SQLite;
+ добавлен плагин с поддержкой изображений в формате WebP;
* плагин RPTImportPlugin обновлен до .NET Framework 4.7.2;
- исправлена ошибка, приводившая к System.IO.FileLoadException при подключении к ClickHouse и MongoDB;
- исправлена форма выбора источника данных, которая открывалась не на первом плане.
```

Версия 2023.2

Новые возможности

Добавлена поддержка Blazor WebAssembly

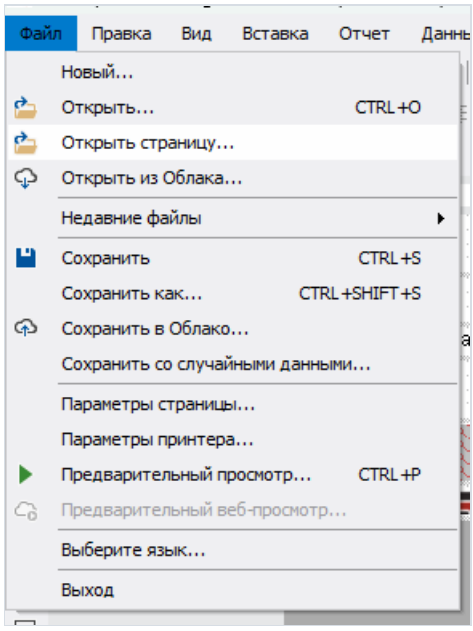
Добавлен пакет FastReport.Blazor.Wasm с поддержкой Blazor WebAssembly для обладателей редакции FastReport .NET Enterprise и выше. Теперь вы можете использовать Razor компоненты для отображения отчёта в вашем WebAssembly приложении. Внимание! На данный момент поддержка Blazor WebAssembly осуществляется в бета-режиме.

```
<WebReportContainer WebReport="WebReport" />
```

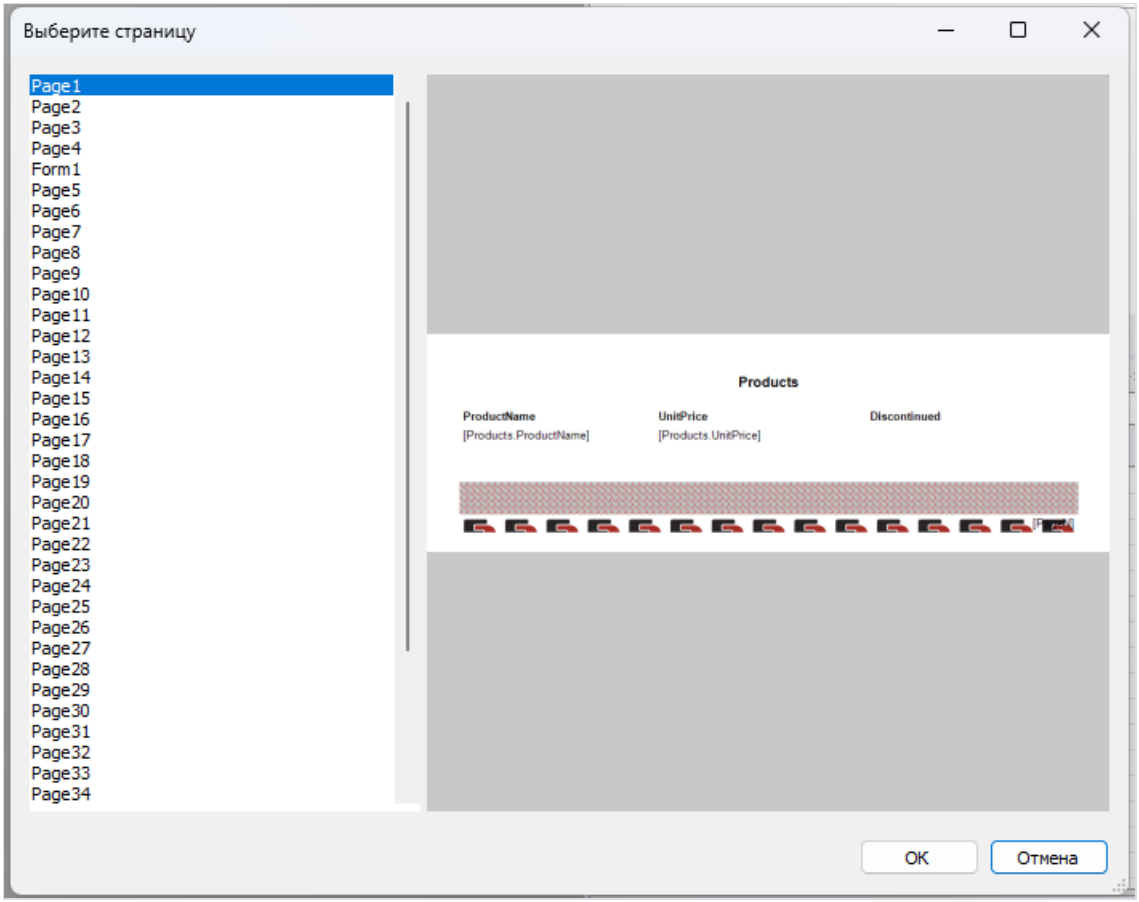
Подробнее читайте в статье.

Возможность открыть страницу другого отчета

В дизайнере появилась возможность открывать и добавлять страницы с диалоговыми формами другого отчета в разрабатываемый отчет. Для этого нужно перейти в меню "Файл" и выбрать пункт "Открыть страницу...".



Далее откроется стандартный диалог выбора файла, в котором можно выбрать отчет. После появится окно со списком страниц и предварительным видом выбранной страницы.

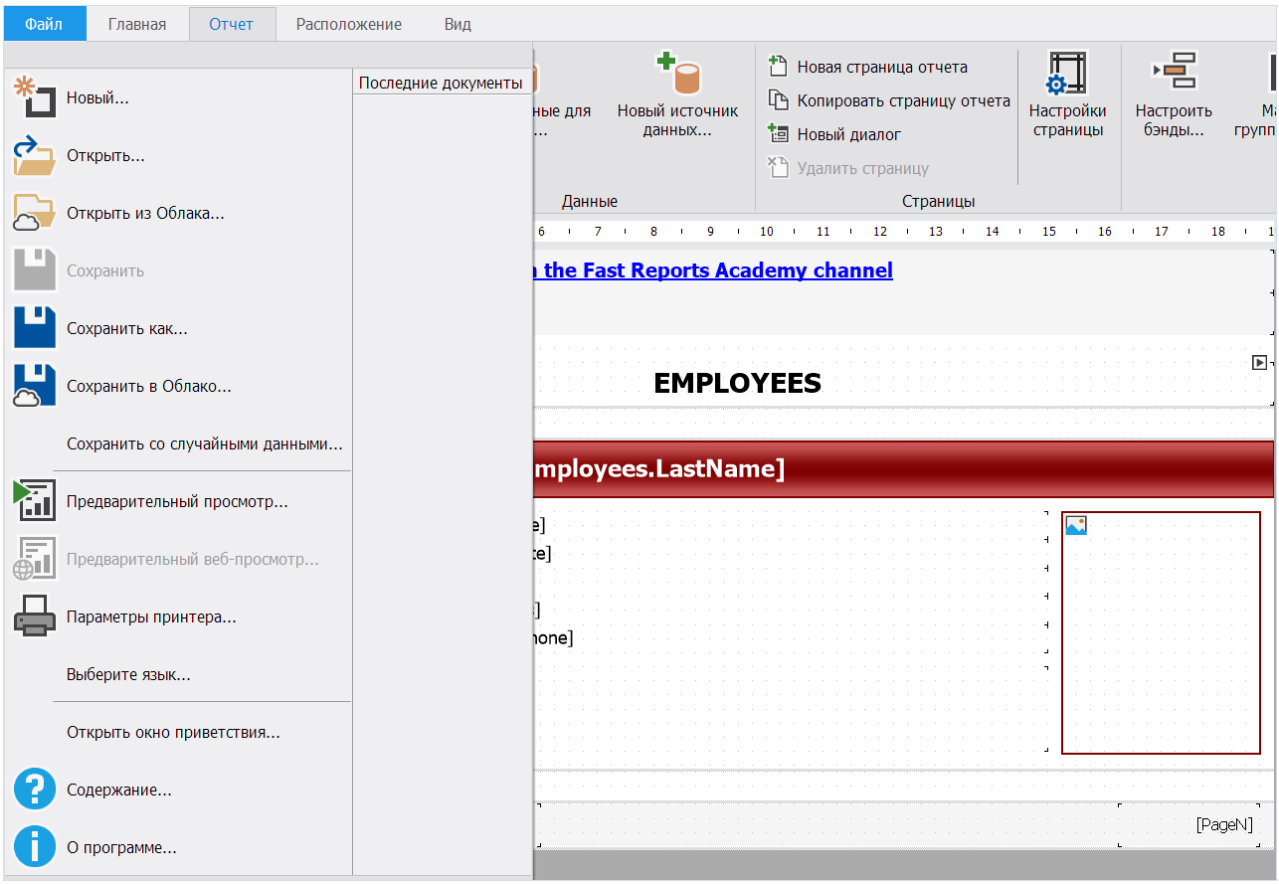


Здесь можно выбрать одну или несколько страниц, которые будут добавлены в текущий отчет. Имена страниц и всех содержащихся на них объектов, будут изменены на уникальные, если в отчете уже есть такие. Это необходимо для исключения ошибок, так как одинаковые имена недопустимы.

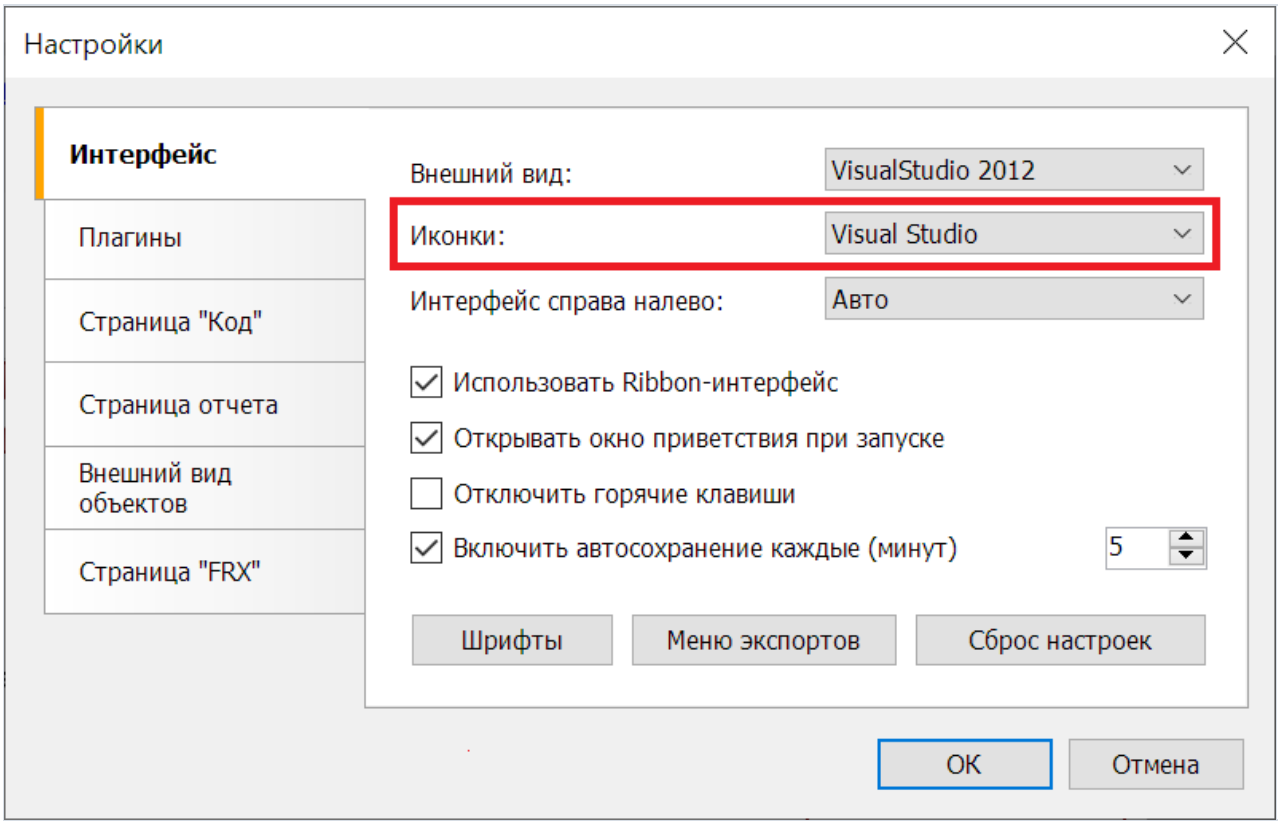
Подробнее читайте в статье.

Новые иконки для Ribbon-интерфейса

Для Ribbon-интерфейса в дизайнере добавлены новые иконки в стиле Visual Studio.



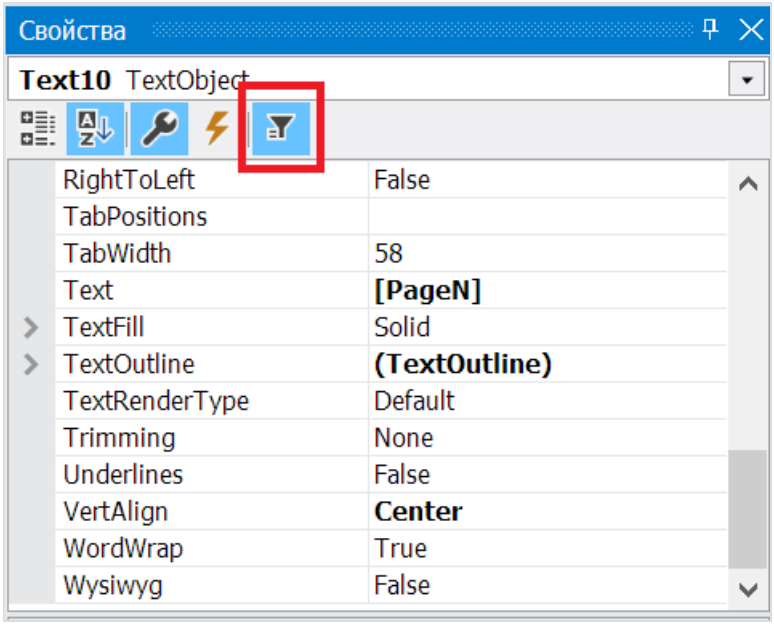
Их можно выбрать в настройках интерфейса.



Потребуется перезапуск дизайнера, чтобы изменения вступили в силу.

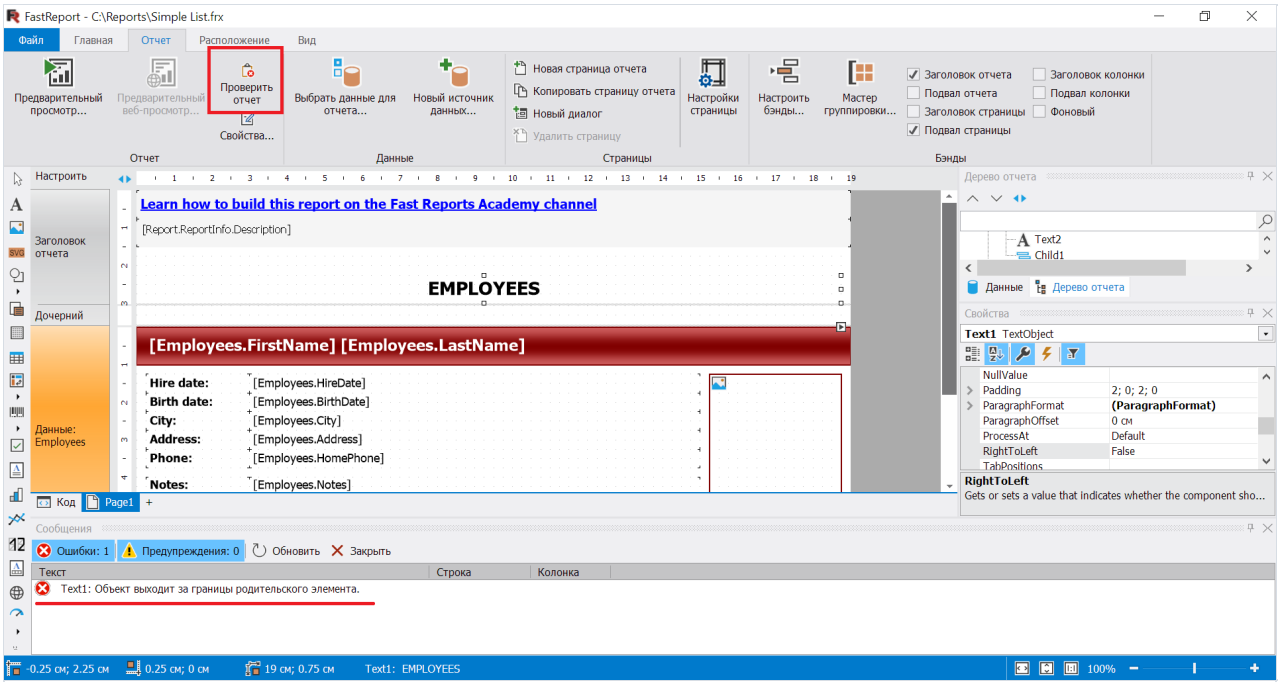
Фильтр в окне свойств

В окне свойств добавлена новая кнопка, которая позволяет включить режим отображения специфичных для объекта свойств. Например, для текстового объекта в этом режиме отображаются свойства Text, Font. И не отображаются общие для объектов свойства, такие как Top, Left, Height и Width.



Изменения валидатора отчетов

Валидатор отчета теперь не работает в фоне, а запускается отдельной кнопкой "Проверить отчет" в меню "Отчет". Кроме этого, окно валидатора удалено, а его сообщения выводятся в окно "Сообщения".



Возможность скрыть строку подключения

Добавлено новое свойство Config.ConnectionStringVisible, которое дает возможность скрыть строку подключения в дизайнере. Может быть использовано для разграничения прав между разработчиком приложения и пользователем отчета. При значении false, пользователь не сможет видеть и редактировать в дизайнере строки подключений.

Изменения WebReport

Добавлена поддержка MemoryCache. По умолчанию на данный момент используется текущий WebReportCache. Включить MemoryCache можно при регистрации сервисов FastReport:

```
services.AddFastReport(options =>
{
  options.CacheOptions.UseLegacyWebReportCache = false;
});
```

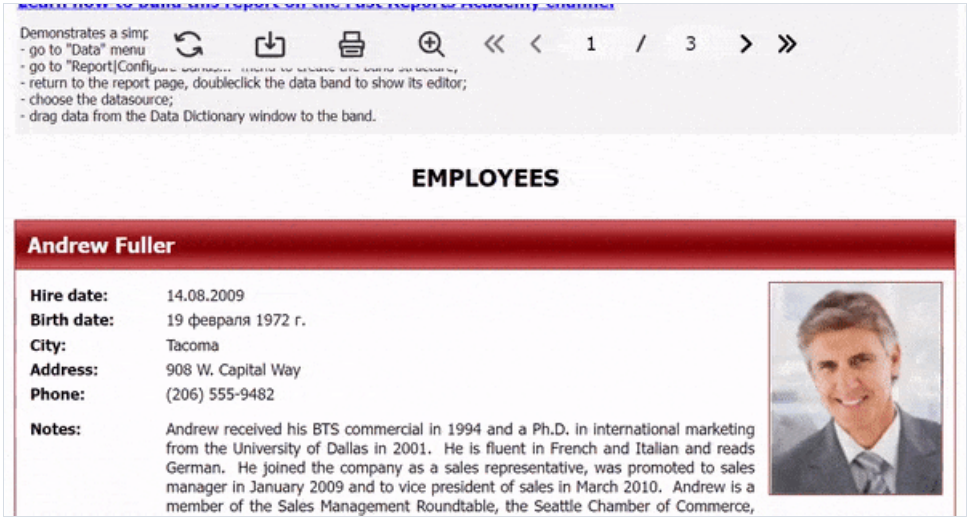
В отличие от встроенного в WebReport кэша, MemoryCache более агрессивно выгружает экземпляры WebReport из памяти из-за отсутствия активности веб-отчёта. Выгрузка происходит после определенного времени, заданного в переменной “CacheOptions.CacheDuration”. Это может помочь в случаях, когда старый кэш по какой-то причине не очищает память.

Добавлена возможность фиксации панели инструментов на экране. Теперь вы можете настроить панель инструментов так, чтобы она всегда оставалась на месте даже при прокрутке страницы. Это удобно при работе с большими отчетами - панель инструментов всегда будет видна.

Чтобы закрепить панель инструментов на экране, необходимо установить следующее свойство:

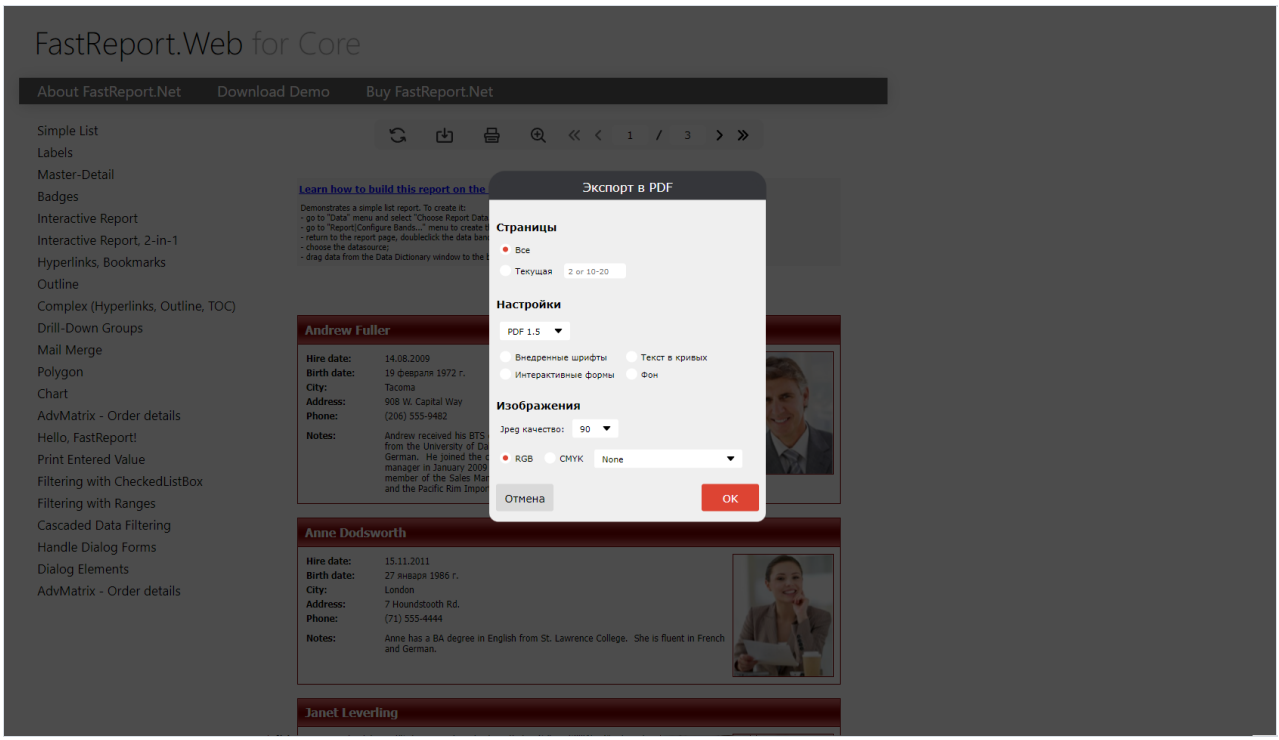
```
webReport.Toolbar.Sticky = true;
```

Теперь панель инструментов всегда будет на экране.

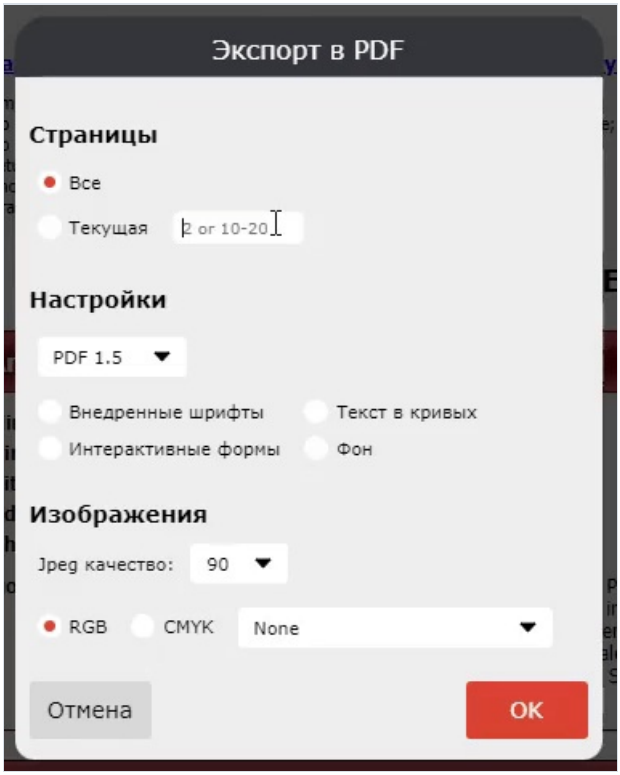


Также была добавлена возможность настройки окна настроек экспорта. Теперь его можно сделать фиксированным на экране и отображать на переднем плане. Для этого необходимо установить следующее свойство:

```
webReport.Toolbar.Exports.PinnedSettingsPosition = true;
```



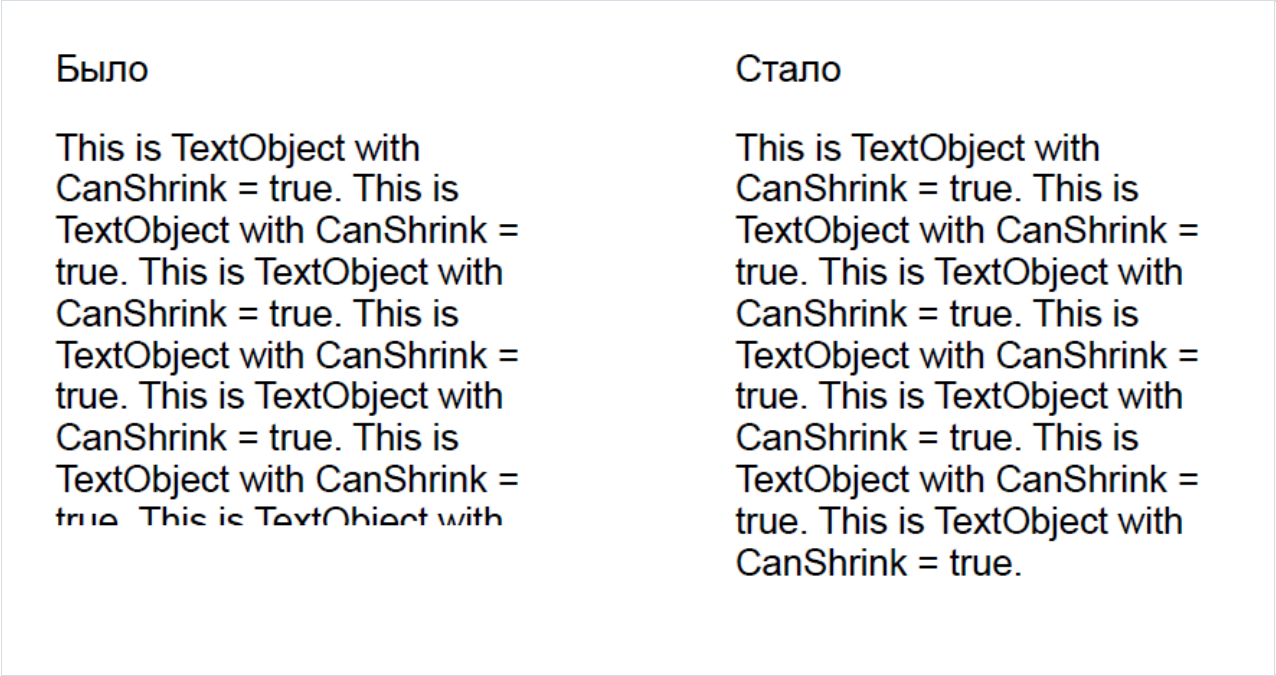
В окно настроек экспорта добавлена валидация для ввода диапазона страниц. Теперь в случае неправильного ввода поле будет выглядеть следующим образом.



Улучшения FastReport.Core.Skia

Улучшена работоспособность пакета FastReport.Core.Skia. Исправлены ошибки экспортов, примеры приведены ниже:

Исправлена отрисовка объектов с CanShrink = true:



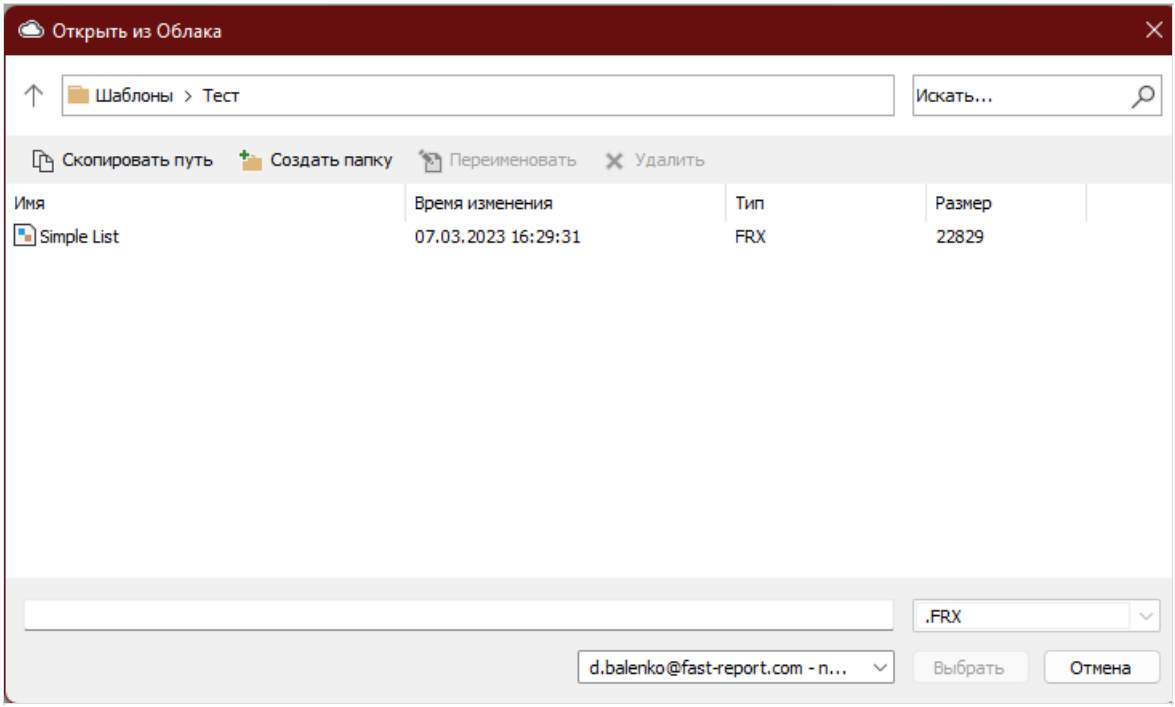
Исправлена отрисовка фона у объектов с прозрачным фоном:

Добавлен стандартный шрифт, который зависит от операционной системы. Теперь, в случае если шрифта из отчёта не будет обнаружено в системе, экспорт не выдаст ошибку, а отрисует отчёт со стандартным шрифтом.

Об остальных исправлениях можно почитать в полном списке изменений.

Обновлен дизайн файлового менеджера МоиОтчеты Облако

Обновлен дизайн окна файлового менеджера для более удобной и интуитивно понятной работы с сервисом. Внесены изменения в расположение элементов интерфейса и цветовую схему, что улучшит общее визуальное восприятие пользователей.



Полный список изменений

```
  [Engine]
+ добавлено свойство Config.ConnectionStringVisible, которое указывает, будут ли отображаться в дизайнере
строки подключения источников данных;
- исправлена ошибка с извлечением процедур в подключении, которое не может содержать процедуры;
- исправлена ошибка, при которой первая колонка страницы всегда выводилась в крайней левой позиции;
- исправлена ошибка, когда свойство GaugeObject.Value устанавливалось равным GaugeObject.Minimum, если
новое значение было больше GaugeObject.Maximum. Теперь оно будет установлено равным GaugeObject.Maximum;

  [Designer]
+ добавлена возможность открыть отчет из МоиОтчеты Облако с помощью списка недавних файлов;
+ добавлено контекстное меню для элементов панели со страницами;
+ для панели со страницами отчета добавлено контекстное меню создания новых страниц и диалоговых форм;
+ добавлены новые иконки в стиле Visual Studio для Ribbon-интерфейса;
+ добавлена кнопка "Синхронизировать" в окно "Дерево отчета";
+ добавлена кнопка "Фильтр" в окне свойств;
+ добавлены HiDPI иконки для Ribbon-интерфейса;
+ добавлена поддержка типов DBNull и Guid для параметров;
* теперь имя прикрепляемого файла можно задать из кода при создании формы экспорта на почту;
* валидатор отчета теперь запускается из меню "Отчет|Проверить отчет". Сообщения валидатора выводятся в
окно "Сообщения";
* изменен интерфейс редактора QR Code;
- исправлена ошибка при нажатии ПКМ на пункте меню "Источники данных";
- исправлена ошибка, при которой не виден чекбокс "Выбрать все" в мастере подключения к данным;
- исправлена ошибка, вызывающая System.NullReferenceException при удалении диалоговой формы;
- исправлено отсутствие API ключа при повторном открытии окна Аккаунт->Сервер, если он был введен в пункт
стандартный сервер;
- исправлен некорректный веб-адрес при попытке предварительного веб-просмотра для кастомного сервера;
- исправлена проблема сворачивания панелей и некорректной смены языка вкладок и панелей при изменении
локализации в Ribbon-интерфейсе;
- исправлена проблема добавления таблиц, которые не были выбраны в мастере подключения;
- исправлена ошибка, вызывавшая исключение System.NullReferenceException при создании подключения к
хранимой процедуре;
- исправлено исключение при ручном вводе недопустимого типа параметра;
- исправлена ошибка, при которой нельзя было установить объекту прозрачный цвет;
- исправлено повторное открытие мастера запросов;

  [Preview]
+ добавлено сообщение об отправке отчета на почту в строке состояния;

  [Exports]
```

```
+ добавлен перенос по словам в ячейках при экспорте в Excel 2007;
- исправлена ошибка, из-за которой текст MSChart размывался после экспорта в HTML;
- исправлены некорректные отступы при экспорте отчета в HTML;
- исправлена ошибка, из-за которой прозрачный фон становился белым при использовании Skia;
- исправлена ошибка с лишней пустой страницей при экспорте, если есть бэнды со свойством Exportable
равным false;
- исправлена ошибка, когда верхний отступ не учитывался при экспорте в послойный HTML;
- исправлена ошибка, из-за которой при отдалении страницы текст выходил за рамки таблицы при HTML
экспорте;

  [WebReport]
+ добавлена поддержка Blazor WebAssembly;
+ добавлена поддержка DI в WebReport.Core/Blazor. Для использования следует вызвать
services.AddFastReport();
+ добавлена поддержка Microsoft.Extensions.Caching.Memory.MemoryCache вместо стандартного
WebReportLegacyCache. Для использования, при регистрации DI контейнера следует использовать
services.AddFastReport(options => options.CacheOptions.UseLegacyWebReportCache = false);
+ реализовано событие ItemCheck в CheckedListBox;
+ добавлена возможность включить отображение тулбара вне зависимости от положения экрана в WebReport с
помощью свойства WebReport.Toolbar.Sticky;
+ добавлена асинхронная версия WebReport.Designer.SaveMethod - WebReport.Designer.SaveMethodAsync;
+ добавлена валидация диапазона страниц в окне настроек экспорта WebReport;
+ добавлено свойство WebReport.Toolbar.Exports.PinnedSettingsPosition. Если оно включено, то контейнер
настроек экспорта будет зафиксирован на экране и отображаться на переднем плане;
- исправлена ошибка, когда при попытке выбрать несколько элементов в ListBox с режимом множественного
выбора, выбирался только один;
- исправлен баг для диалога, при котором он не мог обновиться, когда CheckedBox являлся инициатором
события. В этом случае, добавьте у CheckedBox хотя бы один зависимый объект в свойство DetailControl;
- исправлен баг, при котором в .NET Framework MVC отчет с формой при нажатии на "OK" не убирал форму и не
показывал загрузку отчета;
- исправлена ошибка, из-за которой появлялись лишние страницы при печати;
- исправлена некорректная работа отчета 'Interactive Report' на WebReport.Core;
- исправлена редкая ошибка NullReferenceException в WebReportLegacyCache;

  [Online Designer]
- исправлена ошибка, при которой свойства First Page Source, Other Page Source, Last Page Source и Duplex
не сохранялись при изменении страницы отчета;
- исправлена ошибка, из-за которой превью отчёта не обновлялось до нажатия кнопки "Обновить";

  [.Net Core]
+ компилятор скрипта теперь будет отображать ошибки в зависимости от выбранной локализации, установленной
с помощью FastReport.Utils.Res.LoadLocale() или FastReport.Utils.Config.CompilerSettings.CultureInfo;
- исправлена ошибка, из-за которой неверно отображался текст с CanShrink = True после экспорта в Skia;
- исправлена ошибка, из-за которой неверно рассчитывалась ширина отступа между символами с TextRenderType
= HtmlTags в Skia;
- исправлена ошибка, из-за которой при экспорте с помощью Skia у водяного знака с прозрачностью появлялся
серый фон;
- исправлена ошибка, из-за которой неверно рассчитывалась высота строки таблицы;

  [CoreWin]
- исправлена ошибка при попытке добавить новое подключение;

  [Mono]
+ добавлен элемент управления масштабом в окнах дизайнера и превью;
- исправлена проблема масштабирования PreviewControl;

  [Demos]
+ добавлено демо-приложение ASP.NET Core (Razor pages) под .NET 6.0;
* обновлены демо-приложения для FastReport Core;

  [Extras]
- исправлена ситуация, при которой хост при логауте мог не совпадать с таковым при авторизации;
- исправлен баг, при котором при обновлении истекшей сессии в окне Аккаунт открывался браузер и
запрашивал повторную авторизацию.
```

Версия 2023.1

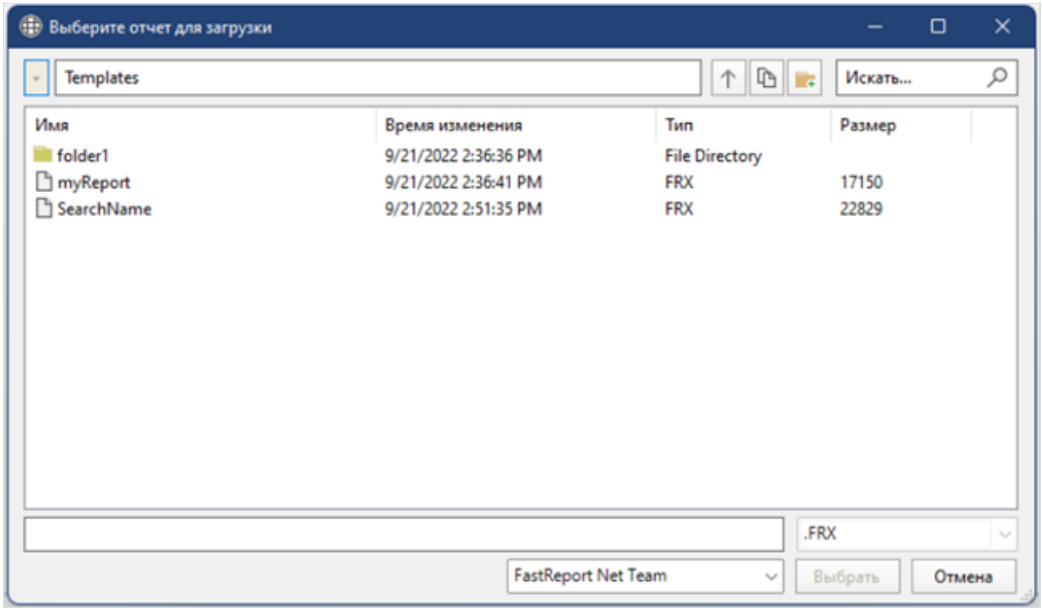
Новые возможности

Интеграция с МоиОтчеты Облако

FastReport .NET, FastReport CoreWin, FastReport Mono начали поддерживать некоторое взаимодействие с МоиОтчеты Облако.

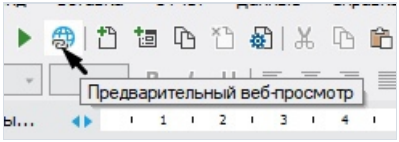
Скачивание и загрузка отчетов

Теперь Вы можете скачать отчет из хранилища МоиОтчеты Облако и работать над ним в дизайнере, или же наоборот – загрузить свои файлы в Облако.



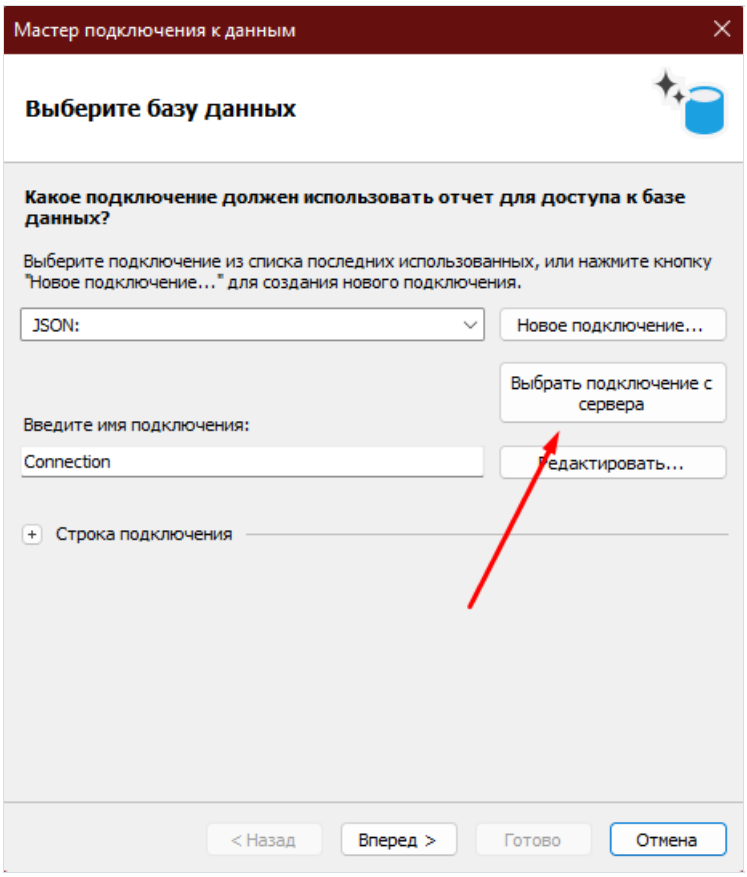
Предварительный веб-просмотр

Помимо стандартного предпросмотра появился и предварительный веб-просмотр. Отчет можно просмотреть таким образом, только если он был открыт из Облака.

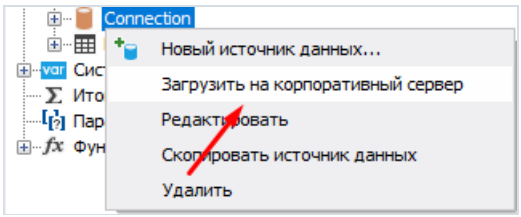


Подключение к источникам данных МоиОтчеты Облако

МоиОтчеты Облако может хранить подключения к источникам данных. С этого момента вам доступна возможность добавить эти источники данных в собственный отчет.



А также появилась возможность загружать подключение в МоиОтчеты Облако.



Чтобы узнать подробнее о новых возможностях, прочитайте эту статью.

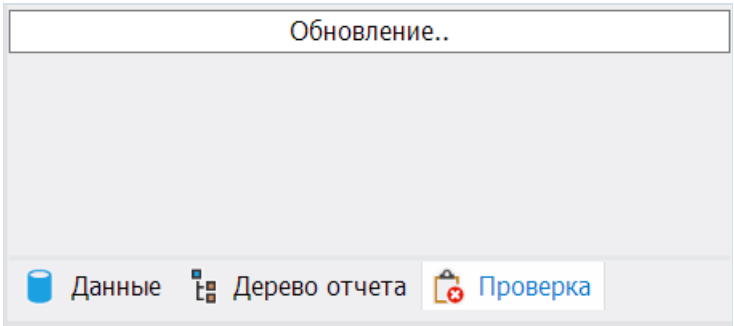
Поддержка .NET 7

Добавлена поддержка .NET 7 для FastReport.Core и FastReport.CoreWin. Эта платформа повышает производительность приложений и добавляет множество новых возможностей для ваших проектов.

Улучшения валидатора отчетов

Повышена скорость работы

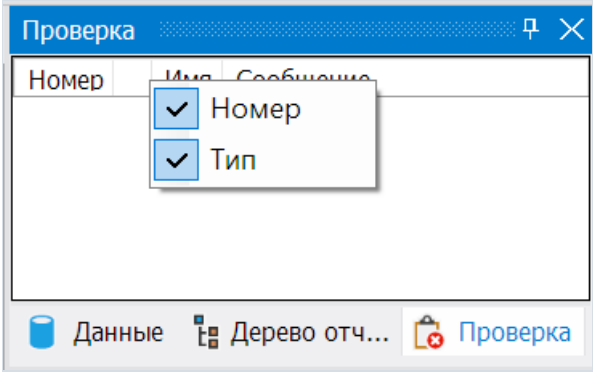
Теперь валидатор отчетов работает в отдельном потоке. А также, значительно оптимизирована скорость его работы. Изменения можно заметить при обработке отчетов с большим количеством ошибок. Пока валидатор проверяет отчет, в окне проверки отображается соответствующее сообщение.



При этом можно редактировать отчет. По завершению работы валидатора, появится таблица с ошибками.

Настройка таблицы валидатора

Для удобства добавлен новый столбец с номерами ошибок. Его отображение можно включить или отключить через контекстное меню таблицы. Таким же образом, можно настраивать отображение столбца с типом ошибки.



Конвертор шаблонов JasperReports

Добавлена возможность конвертировать шаблоны отчетов из JasperReports в шаблоны FastReport .NET. В отчетах JasperReports могут присутствовать объекты, которые не поддерживаются дизайнером FastReport. Эти объекты не будут конвертироваться, либо будут заменены другими таким образом, чтобы построенный отчет был максимально похож на созданный в JasperReports.

Подробнее читайте в статье.

Улучшения и исправления объекта MSChartObject

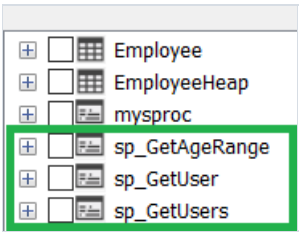
Объект MSChartObject имеет большое количество свойств и настроек. Наиболее часто используемые из них вынесены в редактор объекта. Свойства, недоступные в редакторе, можно изменять с помощью инспектора объектов. Однако с этими свойствами возникала проблема - при их изменении, отчет не считался измененным. В результате сохранение было недоступно.

Чтобы сохранить отчет, нужно было изменить какое-нибудь другое его свойство или объект. Кроме того, при подготовке отчета и после закрытия окна предварительного просмотра, значения указанных свойств сбрасывались на значения по умолчанию.

В новой версии эта ошибка исправлена.

Подключение к хранимым процедурам в MsSQL

Добавлена возможность подключаться к хранимым в MsSQL процедурам. Ранее это было доступно с помощью запроса к базе данных. Теперь к процедурам можно подключиться гораздо удобнее, с помощью интерфейса подключения таблиц базы данных. Они будут отображаться в окне выбора вместе с таблицами.



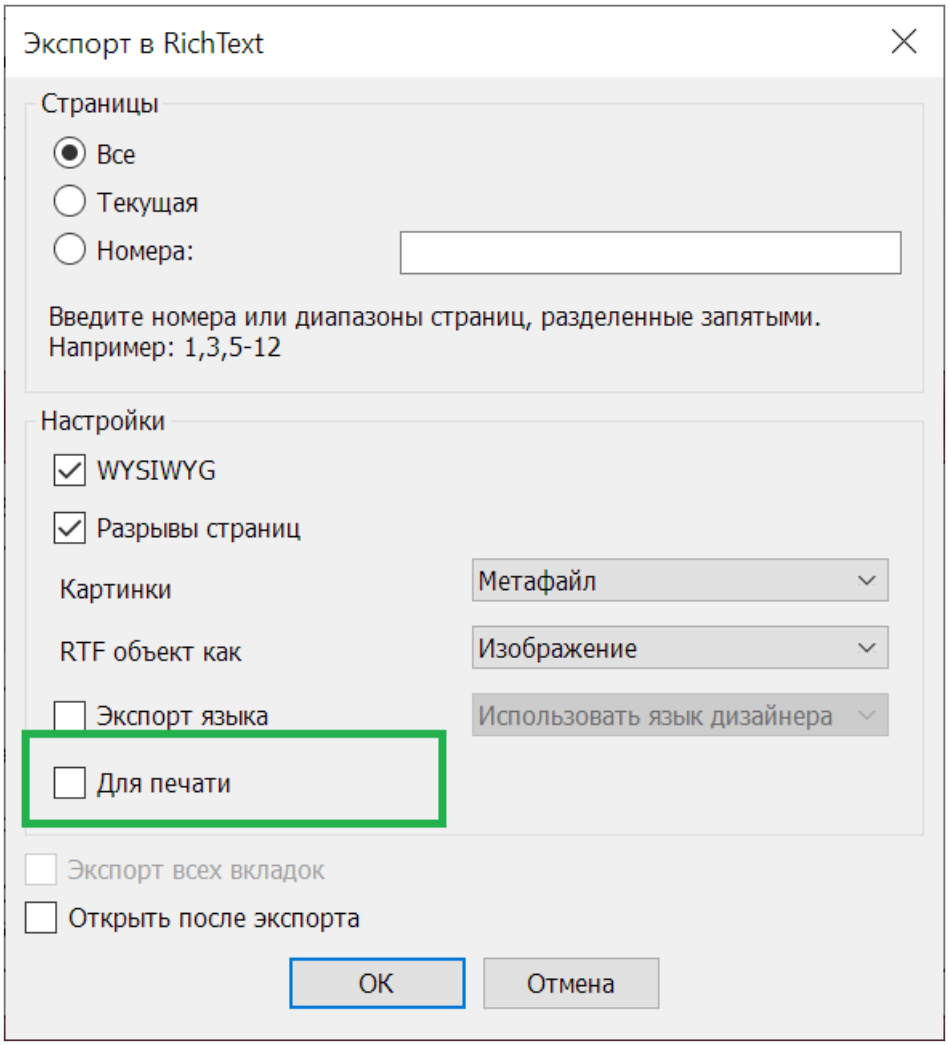
При выборе процедуры будет появляться окно с настройками параметров, если они есть.

Подробнее читайте в статье.

Улучшения экспортов

Опция "Для печати" в RTF экспорте

Добавлено новое свойство PrintOptimized и соответствующая опция в окне экспорта. Включение этой настройки значительно увеличит качество изображения при экспорте. Однако, размер выходного файла тоже увеличится.



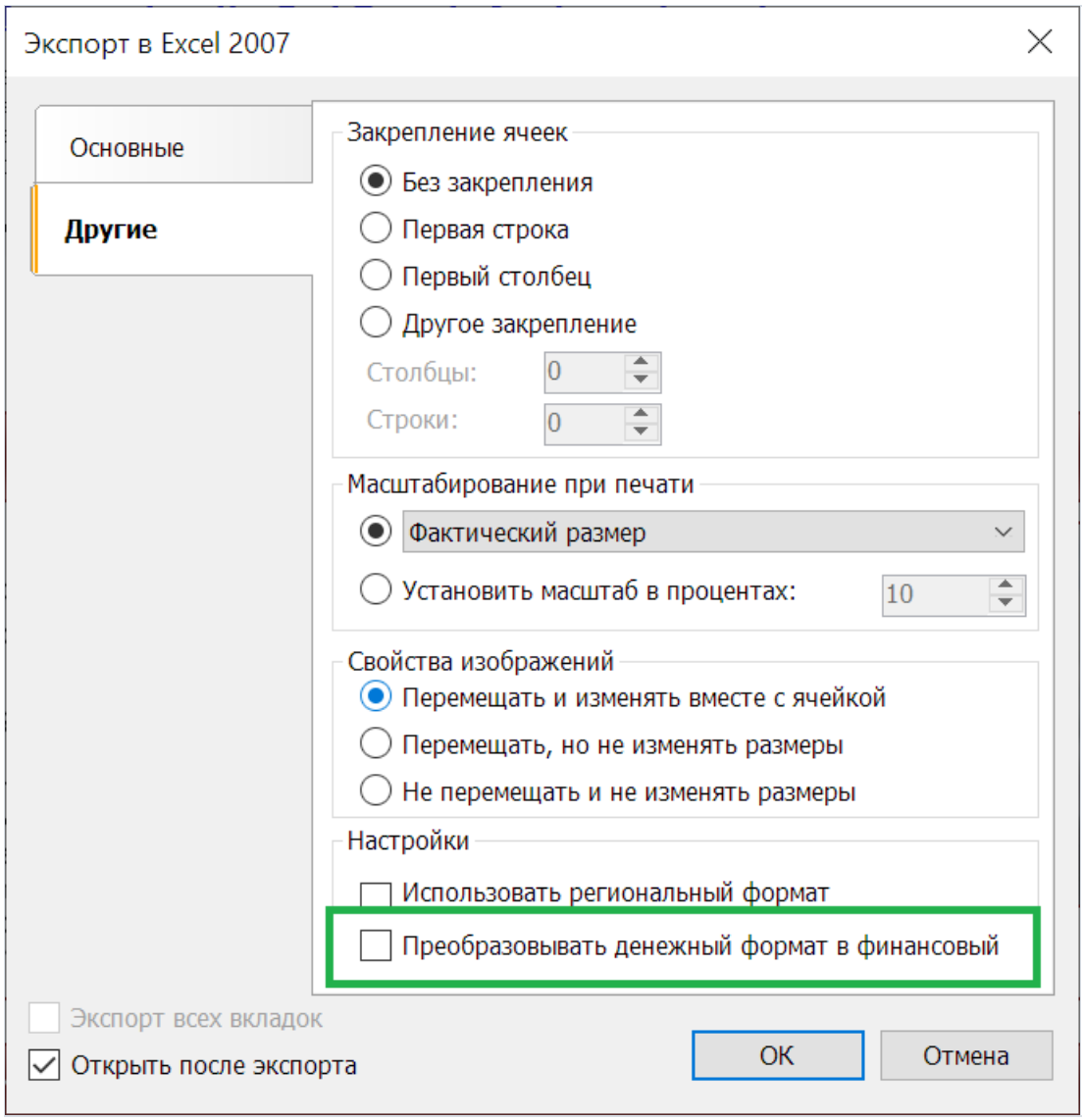
Свойство UseFileStream в Excel 2007 экспорте

Добавлена новая опция UseFileStream для экспорта в Excel 2007. Ее можно использовать только при экспорте из кода в файл. Эта опция полезна при экспорте отчетов с большим количеством страниц (несколько десятков тысяч) в нескольких потоках. Тем самым это позволит избежать ошибок с нехваткой памяти. В остальных случаях использование особого смысла не имеет и не рекомендуется. Пример:

```
Report report = new Report();
Excel2007Export export = new Excel2007Export();
export.UseFileStream = true;
report.Export(export, "report.xlsx");
```

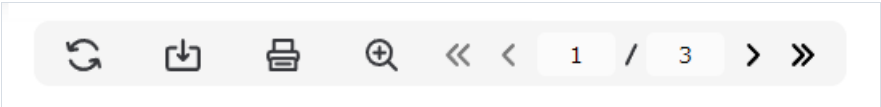
Финансовый формат при экспорте в Excel 2007

Теперь денежный формат данных можно экспортировать как финансовый. Для этого добавлена соответствующая опция в окне экспорта и свойство CurrencyToAccounting.

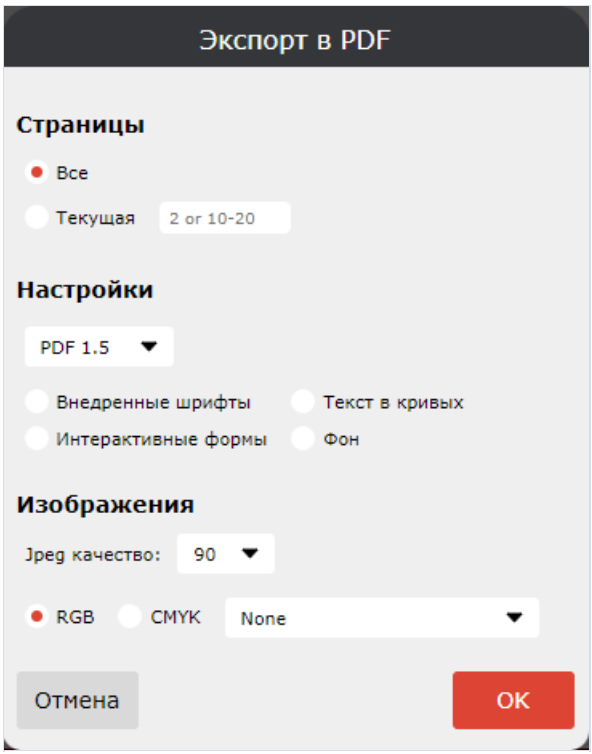


Обновление дизайна WebReport Core/Blazor

Подвергся изменениям внешний вид панели инструментов. Палитра цветов была изменена на более светлую, теперь панель инструментов по умолчанию располагается по центру.



Также изменены формы настроек экспортов. Ранее некоторые свойства задавались с помощью слайдера. Для большего удобства настройка таких свойств теперь задаётся в виде выпадающего списка.



С изменением дизайна панели инструментов сохранилась возможность её кастомизации, поэтому имеется возможность частично вернуть прежнее оформление. Для этого необходимо выставить следующие настройки:

```
var toolbarSettings = new ToolbarSettings
{
    Color = Color.LightGray,
    Roundness = RoundnessEnum.None,
    ContentPosition = ContentPositions.Left,
    IconTransperency = IconTransparencyEnum.None
};

webReport.Toolbar = toolbarSettings;
```

Повышение минимальной версии .NET Framework с 4.0 до 4.6.2

Мы повышаем минимально поддерживаемую версию FastReport .NET до .NET Framework 4.6.2 из-за следующих особенностей:

- поддержка .NET Framework 4.0 давно прекращена;
- наблюдаются проблемы со сборкой исходного кода FastReport в последних версиях Microsoft Visual Studio;
- необходимость для реализации новых API.

Полный список изменений

```
  [Engine]
+ добавлено свойство Report.IsPrepared;
+ добавлен способ отображения текста TextRenderType.Inline;
+ реализован конвертер шаблонов JasperReports;
+ реализовано подключение к хранимым процедурам в MsSQL;
* повышена минимальная версия .NET Framework с 4.0 до 4.6.2;
* получение JSON в источнике данных вынесено в интерфейсную часть;
- исправлена ошибка, приводившая к System.ArgumentException, когда свойство TextObject.FontWidthRatio
равно нулю;
```

- исправлена подсветка текста в RTF парсере;
- исправлена ошибка многократных запросов на получение картинки при использовании URL в ImageLocation;
- исправлена ошибка в функции IsNull;
- исправлена ошибка отрисовки RichObject с выровненными изображениями;
- исправлена ошибка, из-за которой неверно производился расчёт вертикальных расстояний при конвертации RichObject в текст;
- исправлена ошибка в объекте AdvMatrix при обновлении отчета;
- исправлена ошибка с получением строки JSON из JsonTableDataSource;
- исправлена ошибка, приводящая к бесконечному циклу при построении таблицы, если на странице недостаточно места для одной строки;

[Designer]
+ добавлена возможность создания вычисляемого столбца для IEnumerable источников данных;
+ добавлено окно с сообщением о загрузке отчета при открытии файла;
+ добавлен столбец с номерами ошибок в таблице проверки отчета;
+ добавлена возможность скрывать и показывать столбцы с номером и типом ошибки в таблице проверки отчета;
+ добавлено окно, уведомляющее о попытке сохранить отчет, который уже был изменен;
+ добавлена возможность показывать веб-превью отчета, открытого из МоиОтчеты Облако;
+ добавлена возможность взаимодействия с источниками данных в Облаке - скачивание, загрузка, обновление;
* увеличена скорость работы валидатора отчетов;
* кнопка удаления бэнда теперь отключена в ситуациях, когда бэнд удалить нельзя;
* изменено название корневой папки в форме МоиОтчеты Облако, теперь зависит от выбранной локализации;
* теперь в мастере запроса нельзя создать таблицу, если уже есть другая таблица с таким же именем;
- исправлено представление дерева данных с источником данных IEnumerable, столбец которого не добавлялся, если он состоит из значимого типа;
- исправлена ошибка с локализацией кнопки "Удалить" в свойствах отчета на вкладке "Скрипт";
- исправлена ошибка с выделением объекта после щелчка по строке в окне валидации отчета;
- исправлена ошибка, из-за которой не менялся выбранный объект при изменении высоты бэнда мышью;
- исправлена проблема с исключением System.OverflowException при редактировании текстового объекта без редактора;
- исправлена ошибка, вызывающая исключение System.StackOverflowException при копировании форматирования;
- исправлено выделение объекта, расположенного на неактивной странице, при нажатии на строку в окне "Проверка";
- исправлено отображение прогресса обновления списка ошибок в окне "Валидация" при изменении отчета;
- исправлена ошибка с недопустимым значением при изменении цвета линии в редакторе MSChartObject;
- исправлен порядок переключения клавишей "Tab" в формах подключений;
- исправлена ошибка при которой не сохранялись свойства границ осей диаграммы, при их изменении в редакторе;
- исправлены некорректные значения при изменении интервала в полосах на осях в MSChartObject;
- исправлена ошибка, возникающая при удалении бэнда через конфигуратор бэндов, если выбран классический режим отображения бэндов;
- исправлена ошибка, возникающая при нажатии кнопки "Удалить" на форме настройки бэндов, если в отчете отсутствуют бэнды;
- исправлена ошибка, возникающая при удалении бэндов с рабочей области при удержании левой кнопки мыши;
- исправлено восстановление состояния GridControl при закрытии формы редактора столбцов;
- исправлена ошибка, возникающая при нажатии на кнопку "Отмена" в редакторе колонок объекта Grid;
- исправлено отображение метки об изменении отчета при изменении MSChartObject;
- исправлено перемещение столбцов элемента управления сеткой в форме редактора столбцов;
- исправлены ошибки при перетаскивании объектов из дерева отчета на страницы и вкладку "Код";
- исправлены ошибки в окне конструктора запроса, при добавлении таблицы на рабочую область и при создании связей между таблицами;

[Preview]
+ добавлена подсказка для поля "Копия" в форме "Отправить по E-mail";
- исправлено отображение формы печати при увеличении масштабирования дисплея;
- исправлена ошибка, при которой новые экспорты не появлялись в меню;
- исправлен порядок переключения клавишей "Tab" в формах экспортов;
- исправлен левый отступ RichObject;

[Exports]
+ добавлена опция "Для печати" в RTF экспорте;
+ добавлена возможность экспорта денежного формата данных в качестве финансового в Excel 2007 экспорте;
+ добавлена опция UseFileStream для экспорта Excel 2007;
* увеличены формы экспорта для корректного отображения надписей при разных локализациях;
- исправлена ошибка с экспортом линий, нарисованных справа налево или снизу вверх, при экспорте в послойный HTML;
- исправлена ошибка экспорта курсивного шрифта Tahoma в PDF;
- исправлена ошибка, приводящая к System.ArgumentException при экспорте в поток с включенным свойством ImageExport.SeparateFiles;

```
- исправлена ошибка при SVG-экспорте, когда некоторые фигуры прорисовывались дважды;
- исправлена ошибка с экспортом бухгалтерского формата в Excel 2007, при которой не учитывалось
количество десятичных знаков;
- исправлены утечки памяти в экспортах табличного типа;
- исправлена ошибка удаления временного файла при экстренной остановке программы во время экспорта в PDF;
- исправлена ошибка экспорта курсивных и полужирных шрифтов в PDF;
- исправлена ошибка из-за которой не печатался из браузера фон у объектов с заливкой Solid;
- исправлена ошибка с методом set свойства HtmlTemplates.IndexTemplate;
- исправлен экспорт 4-байтовых символов в PDF;
- исправлен множитель высоты строк при экспорте в RTF;
- исправлен множитель высоты строк при табличном экспорте в Word 2007;
- исправлена позиция первого объекта на странице с ненулевым значением при экспорте в Word 2007;
- исправлена ошибка доступа к временному файлу при экспорте в Excel 2007 с использованием свойств
UseFileStream и SplitPages;
- исправлена ошибка с локализацией свойства CurrencyToAccounting в Excel 2007 экспорте;
- исправлена работа кнопок навигации и отображение нумерации страниц в HTML экспорте;
- исправлен подъем и спуск шрифта в PDF-экспорте;

  [WebReport]
* переработано свойство WebReport.ReportPrepared, теперь оно связано с соответствующим свойством отчёта;
* обновлён дизайн WebReport для FastReport.Core.Web и FastReport.Web.Blazor;
- исправлено редкое падение при попытке добавить пустой источник данных в WebReport;
- исправлена ошибка, из-за которой не работал Outline в WebReport.LoadPrepared();
- удалена кнопка обновления при загрузке подготовленного отчёта (.fpx);
- исправлена ошибка, из-за которой неверно высчитывалась ширина табов RichObject в WebReport;
— удалена возможность выбора страниц в параметрах экспорта для одностраничных отчётов;

  [.Net Core]
+ добавлена поддержка .NET 7
+ добавлен метод LoadReport, использующий stream вместо string с именем файла для импорта из StimulSoft;

  [CoreWin]
- убраны лишние компоненты из тулбара Visual Studio;

  [Demos]
* обновлен дизайн демонстрационных отчетов;
- исправлена ошибка масштабирования дочерних окон в новом демонстрационном приложении;

  [Extras]
+ обновлен плагин FastReportBGObjects, добавлена поддержка диаграммы Bubble;
* подключение к ElasticSearch перенесено в отдельный плагин.
```

Версия 2022.3

Новые возможности

Поддержка Skia:

Теперь FastReport.Core поддерживает отрисовку графики и текста с помощью библиотеки SkiaSharp, которая используется вместо System.Drawing.Common + libgdiplus на Linux системах (но также работает и на других операционных системах).

Для этого используются пакеты с суффиксом .Skia:

- FastReport.Core.Skia
- FastReport.Web.Skia

Эта версия ограниченно поддерживает .NET Framework и, в основном, рассчитана на .NET Core/.NET проекты. Для использования в вашем приложении достаточно сменить название пакета FastReport.Core -> FastReport.Core.Skia, а также добавить следующие пакеты для работы на Linux (на Windows и macOS необходимые пакеты добавляются автоматически):

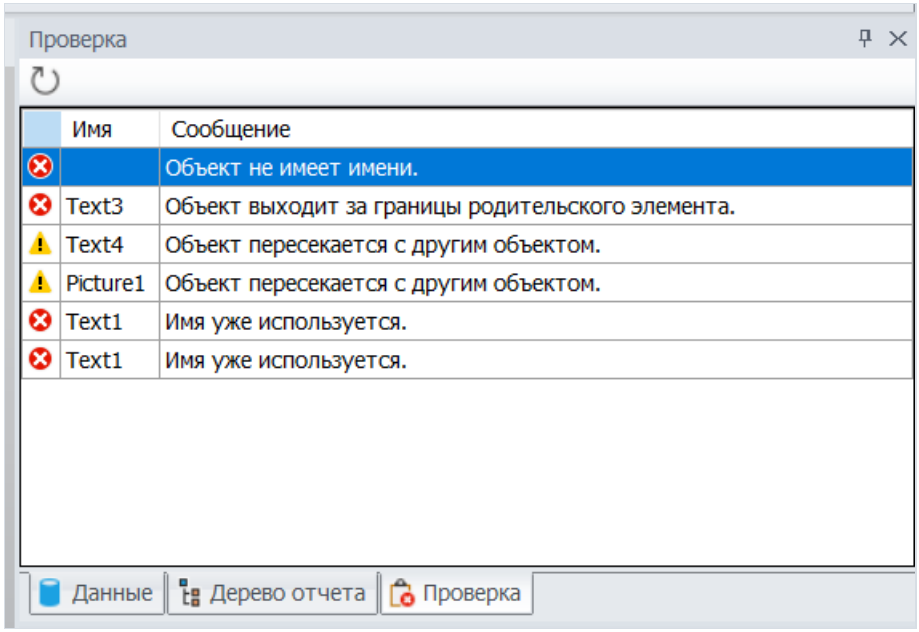
- SkiaSharp.NativeAssets.Linux
- HarfBuzzSharp.NativeAssets.Linux

Подробнее читайте в статье.

Валидатор отчета

В дизайнере отчета добавлена вкладка "Проверка" (справа, рядом с вкладками "Данные" и "Дерево отчета"). Здесь можно проверить шаблон отчета, а также получить список ошибок и предупреждений.

Все это выводится в виде таблицы, в которой указано имя объекта (если оно есть) и описание ошибки. Если выделить строку в таблице, то в дизайнере будет выделен соответствующий объект.



Ошибки и предупреждения могут быть следующих типов: объект без имени, объекты с одинаковым именем, пересекающиеся объекты, объекты с нулевой высотой или шириной, а также объекты, частично или полностью находящиеся за пределами родительского объекта.

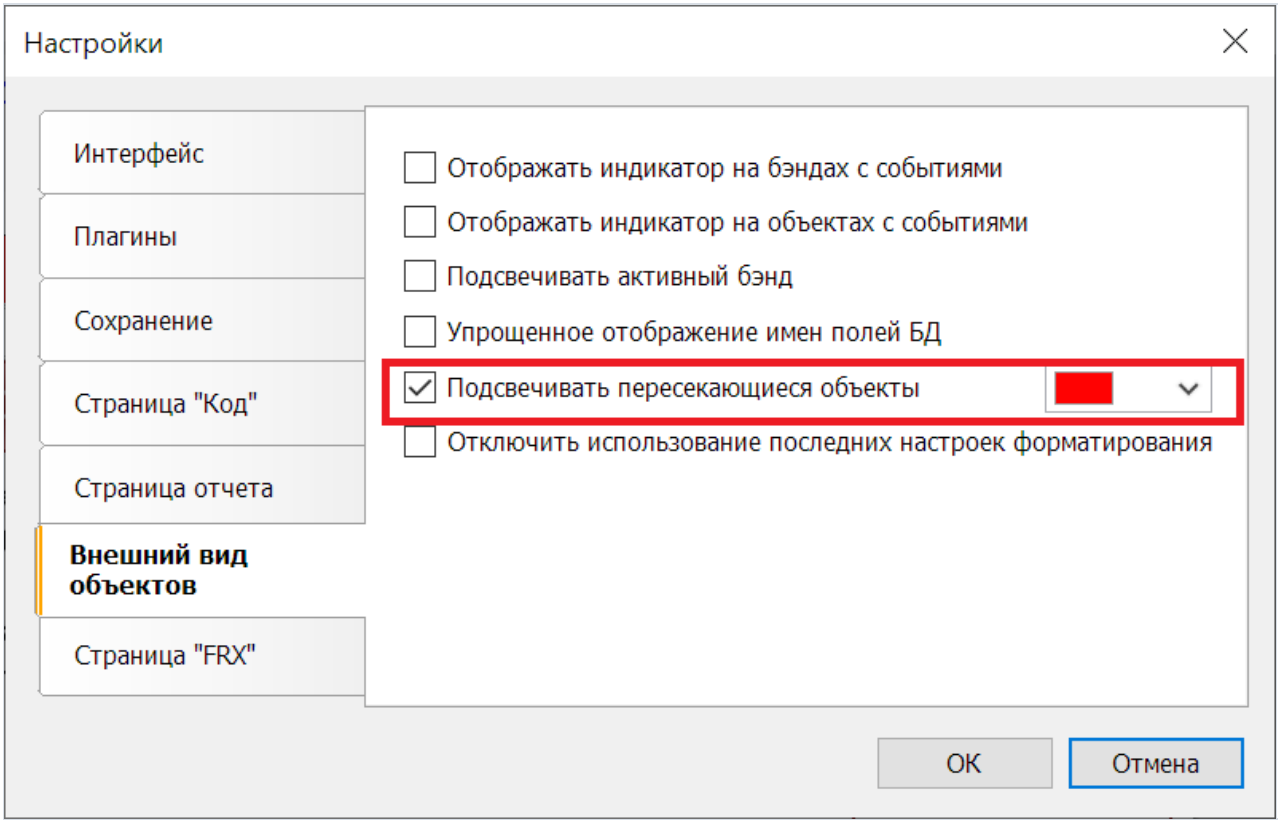
Объекты без имени и объекты с одинаковыми именами, являются критическими ошибками. Они могут приводить к различным ошибкам и даже падению приложения при подготовке отчета. К тому же, без валидатора эти ошибки очень сложно найти.

Пересекающиеся объекты не являются серьезной ошибкой. В некоторых случаях могут быть полезными и использоваться целенаправленно (например, линии или прямоугольники). Пересекающиеся текстовые объекты, в большинстве случаев могут приводить к некорректным экспортам. Особенно в табличных экспортах, например Excel. В результате экспорта будет много лишних ячеек и так далее. С такими объектами надо быть осторожными.

Объекты, частично выходящие за границы родительского объекта (например бэнда или страницы), тоже могут быть полезными в редких ситуациях. Но в большинстве случаев, приводят к ошибкам при подготовке и экспорте отчета.

Объекты, полностью находящиеся за пределами родительского - ошибка серьезная. Найти такие объекты без валидатора, тоже очень трудно.

Пересекающиеся объекты и выходящие за пределы родительских могут выделяться цветом (который теперь можно выбирать), если включена соответствующая настройка в настройках дизайнера.

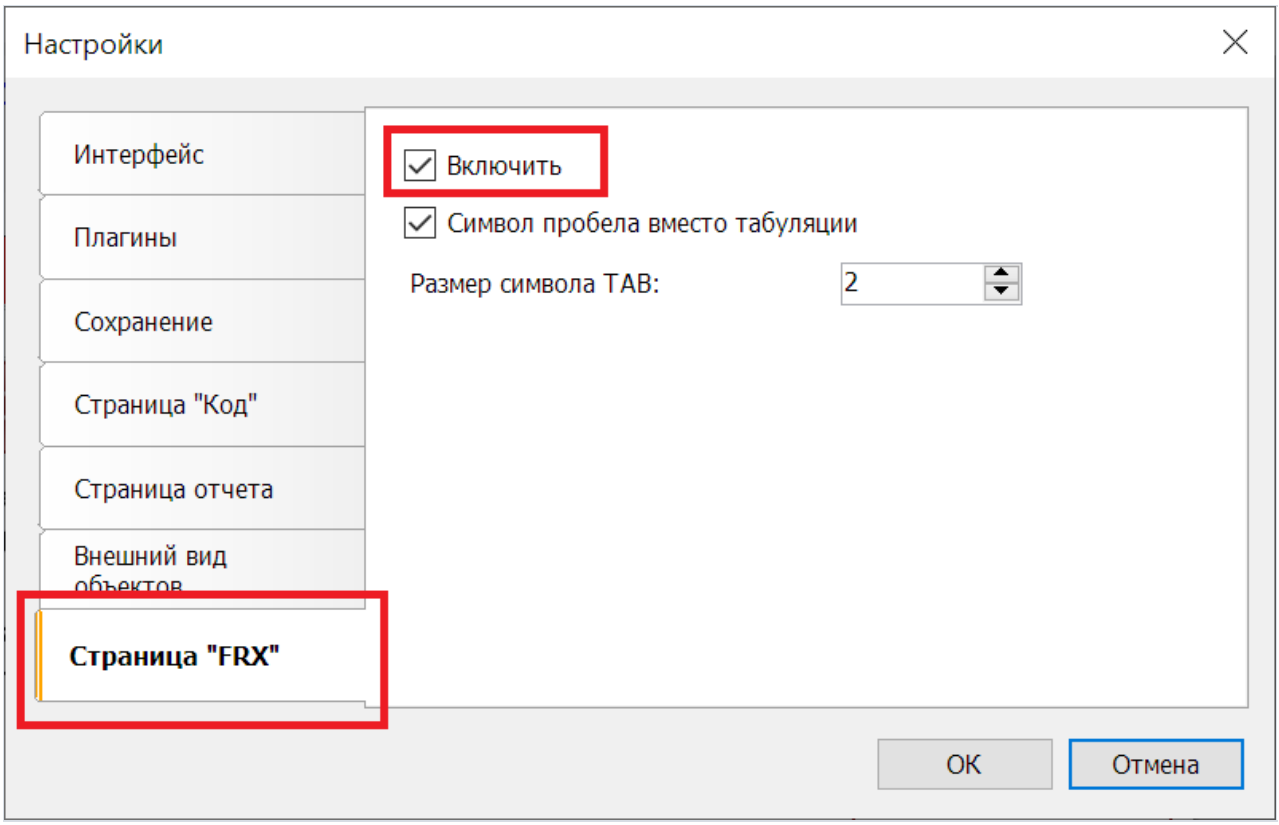


Использовать проверку отчёта совсем не обязательно. Но это может быть полезным, когда ваш отчет работает или выглядит не так как хотелось бы.

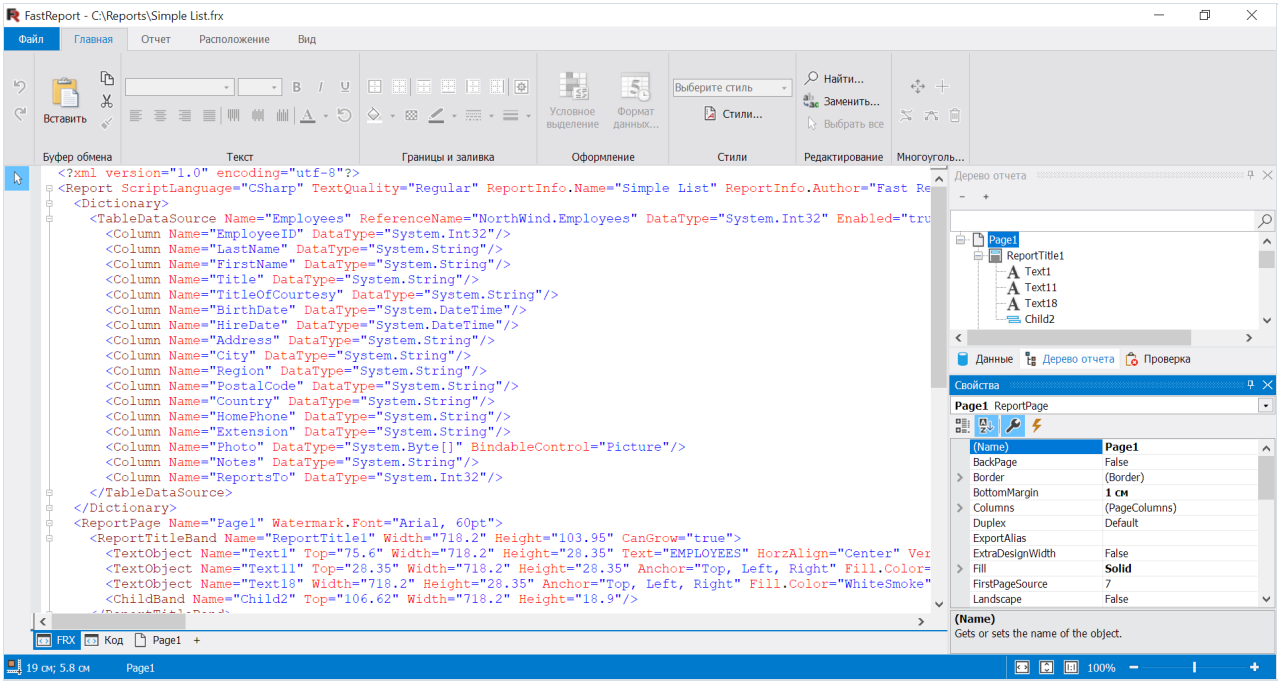
Подробнее читайте в статье.

Редактор FRX

Иногда возникает необходимость редактировать содержимое FRX-файла с помощью сторонних текстовых редакторов. Теперь вы можете это делать непосредственно в дизайнере отчета. Для этого был добавлен редактор FRX формата. По умолчанию он отключен. Чтобы включить редактор перейдите в настройки дизайнера.



В дизайнере отчета, слева от вкладки Code появится вкладка FRX.



Внесенные здесь изменения, будут сразу применяться к отчету и отображаться на его страницах.

Подробнее читайте в статье.

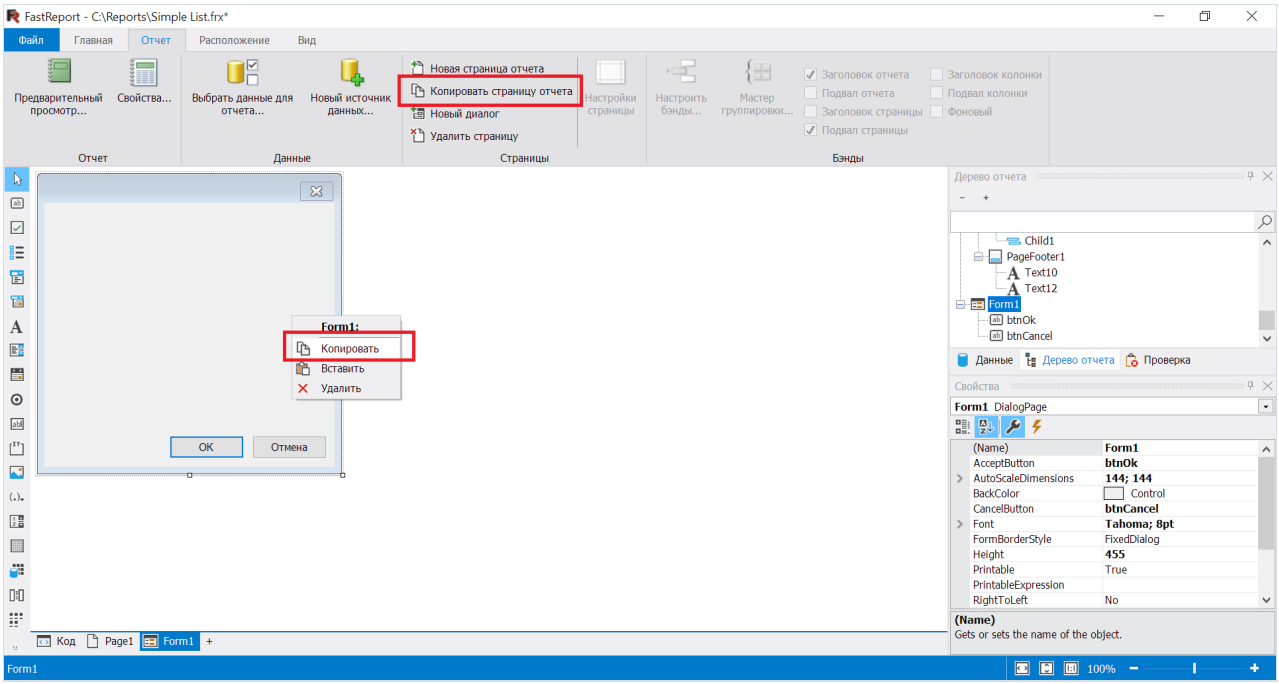
Конвертор отчетов из StimulSoft

Добавлена возможность конвертировать шаблоны отчетов из StimulSoft в шаблоны FastReport .NET. В отчетах StimulSoft могут присутствовать объекты реализации, которые не поддерживаются дизайнером FastReport. Эти объекты не будут экспортироваться, либо будут заменены другими таким образом, чтобы построенный отчет был максимально похож на созданный в StimulSoft. Важно отметить, что импорт кросс-бэндов реализован выносом их содержимого в родительский бэнд.

Подробнее читайте в статье.

Копирование диалоговых страниц

Добавлена возможность копировать диалоговые страницы. Как с помощью контекстного меню диалоговой страницы, так и с помощью кнопки «Отчет -> Копировать страницу отчета».



При копировании создается копия диалоговой страницы с уникальным именем. У всех дочерних объектов тоже будут уникальные имена. Однако обработчики событий у объектов будут те же, что и у исходной страницы. При необходимости нужно создать новые обработчики.

Также, теперь диалоговые страницы можно удалять не только с помощью кнопки «Отчет->Удалить страницу», но и через контекстное меню в редакторе формы и дереве отчета.

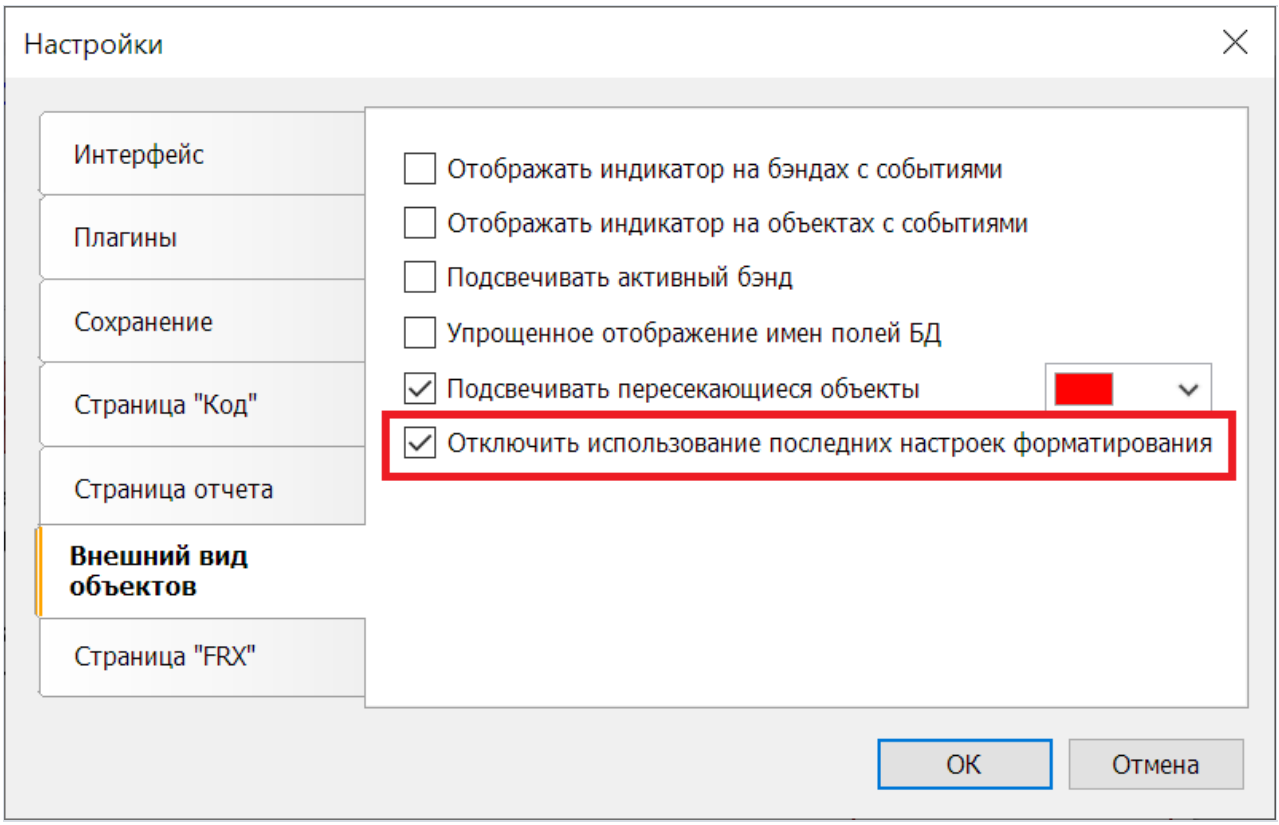
Отключение последних настроек форматирования

При создании объекта в дизайнере его настройки будут применены к следующему созданному объекту того же типа.

Например, если создать текстовый объект, настроить у него размер шрифта, границы, цвет заливки, то следующий текстовый объект будет создан с такими же настройками.

Это удобно, когда нужно создавать несколько объектов с такими же или похожими настройками.

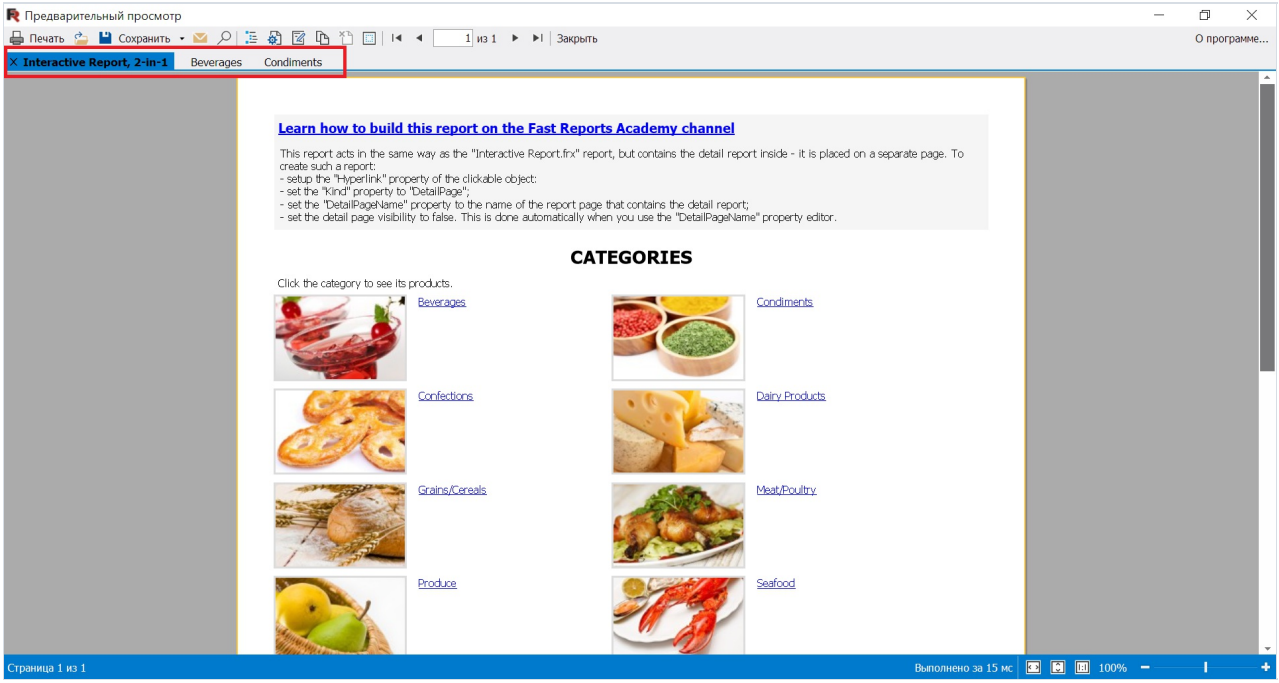
В ситуациях, когда такое поведение дизайнера не нужно его можно отключить в настройках.



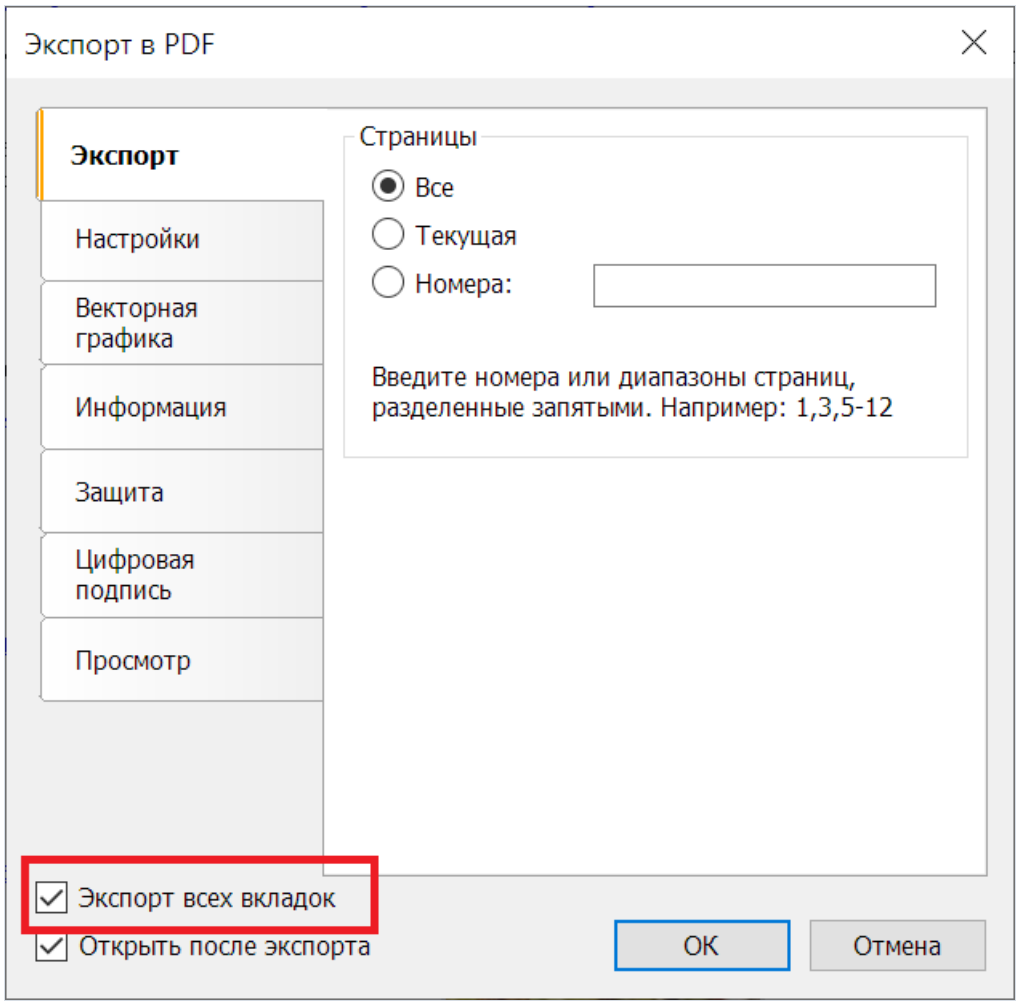
При этом объекты будут создаваться с настройками по умолчанию.

Экспорт всех вкладок

При просмотре интерактивных отчетов можно открывать детальные отчеты в новых вкладках.



Здесь видно три открытые вкладки. Раньше экспортировалась только активная вкладка. Теперь можно экспортировать все вкладки в один файл с помощью новой опции "Экспорт всех вкладок".



Подробное описание связанных сборок и установленных плагинов

Теперь при наведении курсора мыши на dll в списке плагинов (Настройки -> Плагины) и в списке ссылок на сборки (Отчет -> Свойства -> Скрипт), выводится подробная информация с описанием, версией, размером, датой создания и т.д.

Улучшения экспортов

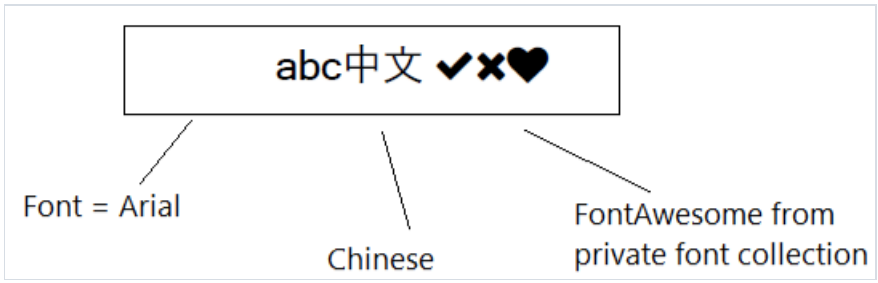
Улучшения в экспорте в PDF:

Linux версия:

- поддержка комплексных языков (арабский, иврит и пр.) в версии Skia.

Все версии:

- поддержка Font Fallback (механизм автоматического выбора шрифта для вывода символов, которые не поддерживаются текущим шрифтом);



- точное позиционирование специальных символов, таких как огласовки и знаки ударения.

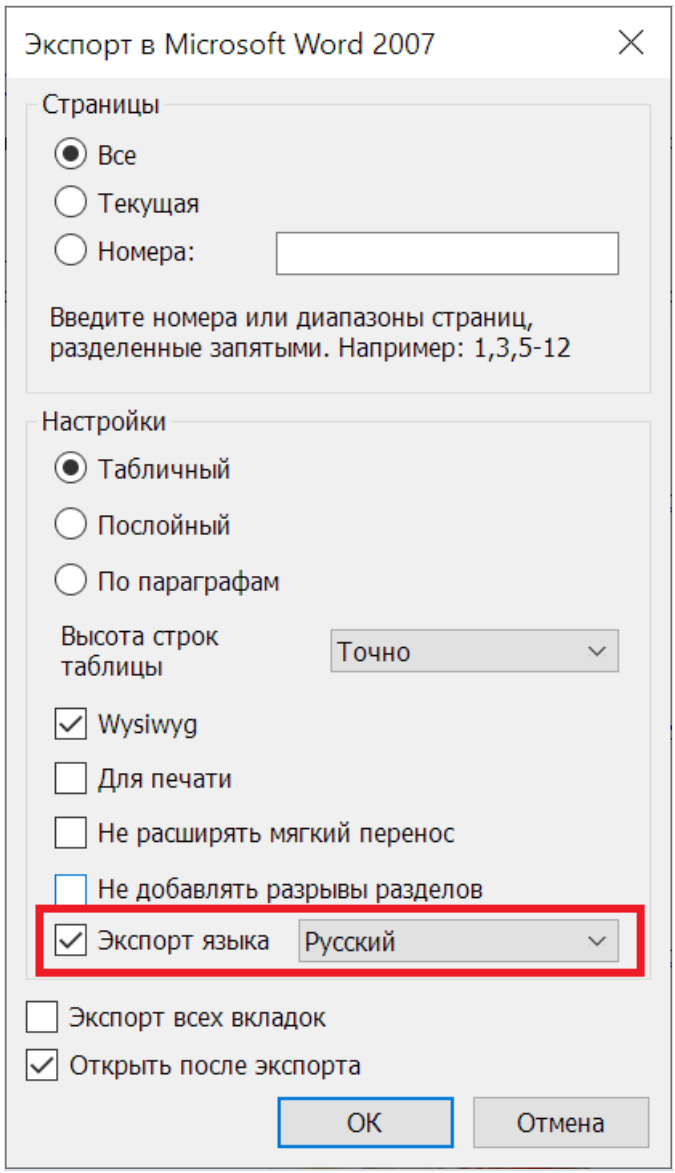
Свойство UseFileStream в PDF-экспорте

Добавлена новая опция UseFileStream для PDF-экспорта. Ее можно использовать только при экспорте из кода в файл. Эта опция полезна при экспорте отчетов с большим количеством страниц (несколько десятков тысяч) в нескольких потоках. Тем самым это позволит избежать ошибок с нехваткой памяти. В остальных случаях использование особого смысла не имеет. Пример:

```
Report report = new Report();
PDFExport export = new PDFExport();
export.UseFileStream = true;
report.Export(export, "report.pdf");
```

Выбор языка при экспорте в Word, PowerPoint, Rich Text, OpenOffice Write и OpenOffice Calc

Теперь в этих экспортах можно выбирать язык документа. По умолчанию используется язык, выбранный в дизайнере.



Также добавлена опция "Показать линии сетки" при экспорте в Excel 2007.

Полный список изменений

[Engine]

- добавлен конвертор отчетов из StimulSoft;
- добавлено изменение имени источника данных JSON в выражениях при его переименовании;
- добавлена конвертация свойства PaperSize при конвертации отчетов из StimulSoft;
- добавлена проверка существования связанных сборок при конвертации отчетов из StimulSoft;
- добавлено свойство PrintOnParent у объектов Table и Matrix;
- добавлена загрузка параметров отчета при конвертации отчетов из RDL;
- добавлена загрузка вложенных отчетов при конвертации отчетов из RDL;
- добавлена возможность сохранить данные JSON подключения с помощью свойства StoreData;
- улучшено быстродействие в отчетах, содержащих большое количество объектов;

- изменен текст исключения при вычислении и форматировании выражения, если e.InnerException null;
- при загрузке RDL отчетов, ширина страницы будет равна ширине секции только в случае, если ширины страницы нет;

- исправлено вычисление длины кодировки DataMatrix C40 и текста;
- обработано исключение System.ComponentModel.Win32Exception, при печати с отключенным "Диспетчером печати";
- исправлено скрытие границы изображения при печати с автоматическим размером;

- исправлена ошибка переполнения стека при подготовке отчета с дочерним бэндом подвала страницы и включенной у него опцией начать новую страницу;
- исправлена ошибка, при которой не передавался текущему отчету путь к базовому отчету на Unix ОС;
- исправлена ошибка с созданием подотчета и страницы с одним именем при конвертации отчетов из StimulSoft;
- исправлена ошибка с некорректными именами при конвертации отчетов из StimulSoft;
- исправлена ошибка с TotalPages в Page.VisibleExpression, которая приводит к исключению при выключенном двойном проходе;
- исправлена ошибка, при которой бэнд может расти за пределы страницы;
- исправлена ошибка, при которой объекты могут расти за пределы бэнда или ContainerObject;
- исправлена ошибка обратного отступа при трансляции RTF в объекты отчёта;
- исправлен межстрочный интервал для текста транслированного из RichObject;
- исправлена ошибка свойства ConnectionString в классе JsonDataSourceConnectionStringBuilder, когда информация приходила без заголовков запроса;

[Designer]

- добавлен валидатор отчета, который помогает находить некорректные объекты (повторяющиеся имена, отрицательные размеры и т.д.);
- добавлен редактор для свойства RichObject.Text;
- добавлен редактор FRX в дизайнере отчетов;
- добавлено подробное описание связанных сборок и установленных плагинов;
- добавлена возможность копировать диалоговые страницы;
- добавлена возможность удалять диалоговые страницы через контекстное меню;
- добавлена возможность отключить использование последних настроек форматирования при создании объектов;
- добавлена интеграция с FastReport.Id;
- добавлен вызов онлайн-документации в дизайнере отчетов;
- добавлен мастер для визуализации контрольных идентификационных знаков;
- добавлены всплывающие подсказки о правых и нижних отступах для направляющих и объектов в дизайнере;
- добавлена возможность выбирать цвет подсветки пересекающихся объектов в дизайнере;
- добавлена возможность подключения баз Access 2007;

- изменен внешний вид формы редактора подключения к ElasticSearch;
- изменены текстовые поля в CISWizardForm с единицами измерения на текстовые поля, поддерживающие только числа;

- исправлена ошибка, приводящая к System.NullRefereceException при создании вычисляемого столбца для вложенной таблицы JSON;
- исправлена ошибка, приводящая к System.FormatException при отрисовке подписей карт;
- исправлена ошибка, приводящая к System.NullReferenceException, при нажатии кнопки "Вставить" в контекстном меню диалоговых страниц;
- исправлена ошибка масштабирования элементов управления зумом дизайнера в режиме HiDPI при запуске из старого демонстрационного приложения;
- исправлено открытие формы сохранения изменений после сохранения всего отчета;
- исправлены не масштабируемые элементы в окне приветствия;
- исправлена подсветка пересекающихся графиков;
- исправлено исключение при переименовании таблицы JSON;
- исправлен UpdateStatusBar в DialogWorkspace;
- исправлена ошибка локализации кнопки "Учётная запись" в меню "Файл";
- исправлена потеря фокуса выбранного объекта при изменении его свойств;
- исправлена ошибка при которой не происходило переключение на страницу "Код" после добавления обработчика событий;

[Preview]

- реализован экспорт всех открытых вкладок;
- исправлена ошибка, приводившая к System.NullReferenceExteption при подготовке отчета с RichObject в системе без принтеров;
- исправлена ошибка в объекте MSChart в режиме HiDPI;

[Exports]

- добавлен экспорт локали в Word, PowerPoint, Rich Text, OpenOffice Write и OpenOffice Calc;
- добавлено шифрование пароля сертификата цифровой подписи в PDF-экспорте при его сохранении;
- добавлена опция "Показать линии сетки" при экспорте в Excel 2007;
- добавлен экспорт типов данных в DBF;
- добавлено новое свойство в экспорт SVG PrefixStyle, которое позволяет задать префикс для всех стилей внутри SVG экспорта;
- добавлена опция "Использовать системное форматирование данных" в Excel 2007 экспорте;
- добавлено свойство PDFExport.UseFileStream, которое позволяет экспортировать огромные отчеты на системах с небольшим объемом оперативной памяти без System.OutOfMemoryException;
- установлена UTF-8 в качестве кодировки по умолчанию для экспорта DBF;
- исправлено неправильное масштабирование изображений в послойном HTML-экспорте при включенном высоком качестве SVG и увеличении более 1;
- исправлена ошибка, приводившая к System.IndexOutOfRangeException при экспорте шрифта без кернинга в PDF;
- исправлена ошибка масштабирования изображений при послойном HTML-экспорте;
- исправлена ошибка, приводящая к System.NullReferenceException при экспорте отчета с пустой страницей в Word 2007;
- исправлена утечка памяти в экспорте PDF с некоторыми CJK шрифтами;
- исправлен баг, при котором SVG картинка не поворачивалась на заданный угол в HTML/Blazor экспорте;
- исправлен повторный рендеринг ячейки таблицы при экспорте в SVG;
- исправлен некорректный стиль страницы при печати из браузера для табличного HTML экспорта;
- исправлено исключение, которое возникало при экспорте объекта с отрицательными размерами в HTML формат;
- исправлена ошибка экспорта в PDF при Compressed = false;
- исправлена некорректная запись свойства border-collapse в табличном HTML-экспорте;
- исправлена ошибка Excel-экспорта, при которой заливка в выходном файле не менялась с первого раза;
- исправлена ошибка экспорта водяного знака в PostScript;
- исправлена ошибка масштабирования шрифта при экспорте в PDF;
- исправлена ошибка, при которой текстовый объект с HtmlTags, экспортированный в RTF, не изменялся тегами
  , ' ;

[WebReport]

- свойства OnlineDesginer перенесены в WebReport.Designer с сохранением обратной совместимости;

- исправлен баг, когда событие "CheckedChanged" RadioButton не выполнялось;
- исправлено некорректное масштабирование Dialog компонентов в Blazor;
- исправлен баг с некорректным размером шрифта при экспорте в Excel;
- исправлен баг в Blazor, при котором шрифт текстового объекта со свойством TextRenderType = HtmlParagraph всегда был стандартным;

[.NET Core]

- исправлен некорректный поиск public-методов в скрипте отчёта;
- исправлена проблема создания файла fontlist на Azure;

[CoreWin]

- для Visual Studio исправлено поведение компонентов WinForms в Toolbox (Design-Time);
- исправлен некорректный запуск браузера при клике по ссылке в CoreWin;
- для FastReport.CoreWin исправлены отчёты со скриптом, которые используют WinForms API;

[Demos]

- добавлена возможность сменить локализацию нового демо-приложения без его перезапуска;
- добавлено демо-приложение на React с FastReport.Core;
- исправлено положение одной из диаграмм в Chart.frx;

[Plugins]

- реализовано подключение к ЛИНТЕР;
- реализовано подключение к Cassandra;
- обновлён RPTImportPlugin;

[Extras]

- для пакетов FastReport.Net* добавлены библиотеки FastReport.Web (для .NET Framework) и FastReport.VSDesign;
- добавлена возможность импортировать отчёты с помощью потоков;

[Service]

- исправлена некорректная версия FastReport.Compat в пакетах FastReport.Net.

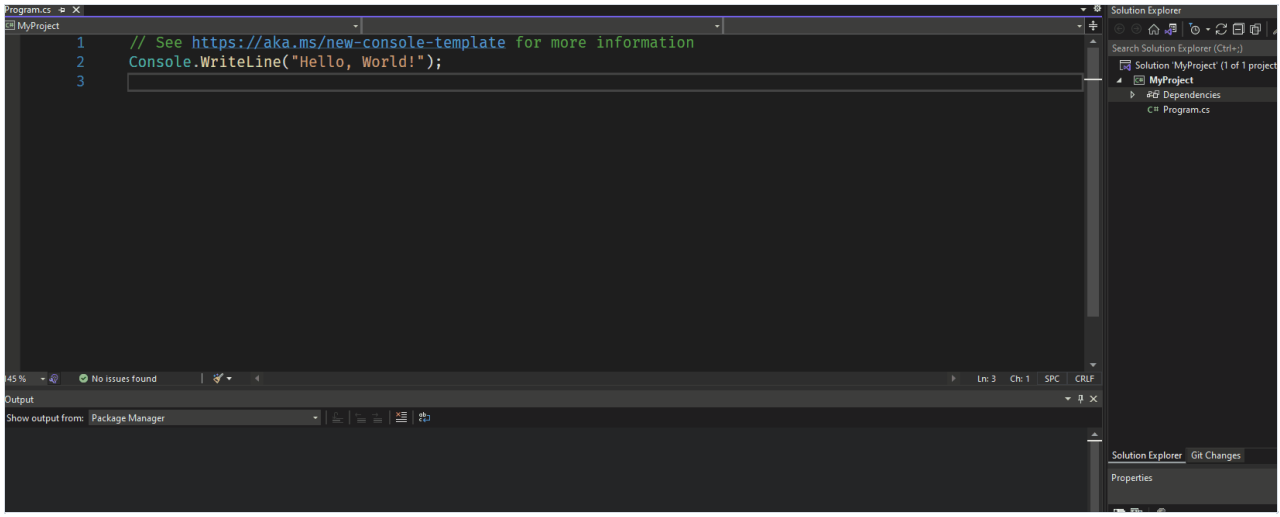
Версия 2022.2

Новые возможности

Fast Reports NuGet-сервер

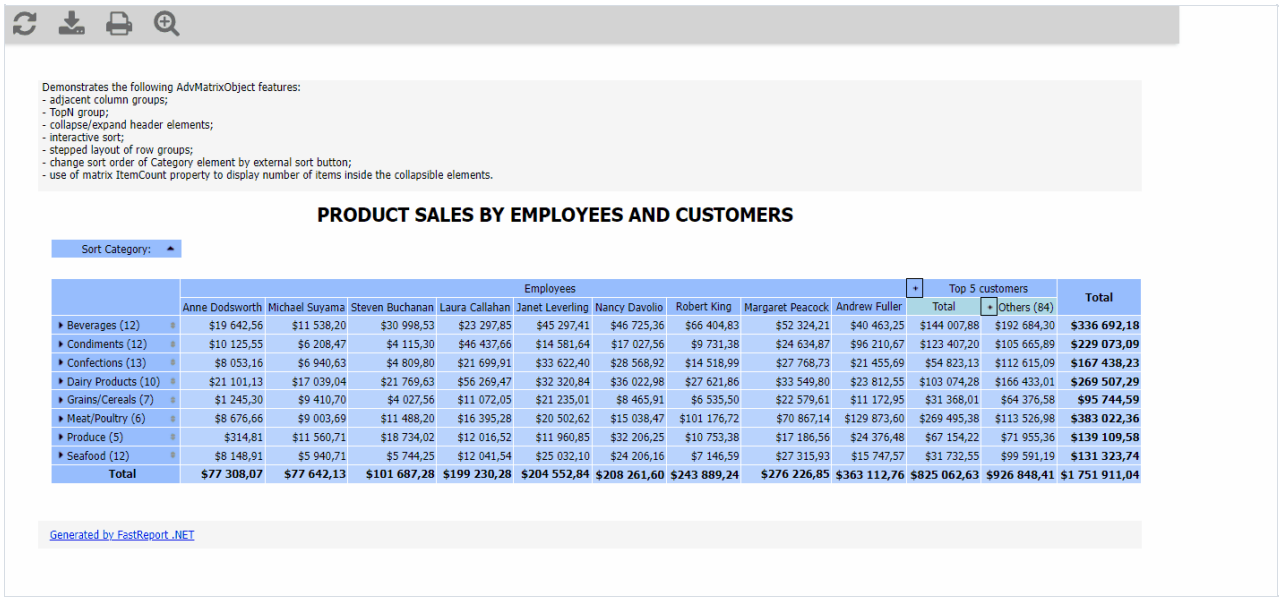
Недавно мы запустили собственный NuGet-сервер - хранилище лицензионных продуктов Fast Reports для пользователей. Теперь вы можете удобно загружать последние версии наших компонентов на любой операционной системе.

Подробнее читайте в статье.



Добавлена поддержка интерактивности объекта "Улучшенная матрица" в WebReport:

Во всех WebReport (.NET Framework, .NET Core, Blazor Server) появилась поддержка интерактивности для новой Улучшенной матрицы. Теперь в браузере корректно работают кнопки сворачивания и сортировки.



Demonstrates the following AdvMatrixObject features:
- adjacent column groups;
- TopN group;
- collapse/expand header elements;
- interactive sort;
- stepped layout of row groups;
- change sort order of Category element by external sort button;
- use of matrix ItemCount property to display number of items inside the collapsible elements.

PRODUCT SALES BY EMPLOYEES AND CUSTOMERS

Sort Category:

	Employees									Top 5 customers		Total
	Anne Dodsworth	Michael Suyama	Steven Buchanan	Laura Callahan	Janet Leverling	Nancy Davolio	Robert King	Margaret Peacock	Andrew Fuller	Total	Others (84)	
Beverages (12)	$19 642,56	$11 538,20	$30 998,53	$23 297,85	$45 297,41	$46 725,36	$66 404,83	$52 324,21	$40 463,25	$144 007,88	$192 684,30	**$336 692,18**
Condiments (12)	$10 125,55	$6 208,47	$4 115,30	$46 437,66	$14 581,64	$17 027,56	$9 731,38	$24 634,87	$96 210,67	$123 407,20	$105 665,89	**$229 073,09**
Confections (13)	$8 053,16	$6 940,63	$4 809,80	$21 699,91	$33 622,40	$28 568,92	$14 518,99	$27 768,73	$21 455,69	$54 823,13	$112 615,09	**$167 438,23**
Dairy Products (10)	$21 101,13	$17 039,04	$21 769,63	$56 269,47	$32 320,84	$36 022,98	$27 621,86	$33 549,80	$23 812,55	$103 074,28	$166 433,01	**$269 507,29**
Grains/Cereals (7)	$1 245,30	$9 410,70	$4 027,56	$11 072,05	$21 235,01	$8 465,91	$6 535,50	$22 579,61	$11 172,95	$31 368,01	$64 376,58	**$95 744,59**
Meat/Poultry (6)	$8 676,66	$9 003,69	$11 488,20	$16 395,28	$20 502,62	$15 038,47	$101 176,72	$70 867,14	$129 873,60	$269 495,38	$113 526,98	**$383 022,36**
Produce (5)	$314,81	$11 560,71	$18 734,02	$12 016,52	$11 960,85	$32 206,25	$10 753,38	$17 186,56	$24 376,48	$67 154,22	$71 955,36	**$139 109,58**
Seafood (12)	$8 148,91	$5 940,71	$5 744,25	$12 041,54	$25 032,10	$24 206,16	$7 146,59	$27 315,93	$15 747,57	$31 732,55	$99 591,19	**$131 323,74**
Total	**$77 308,07**	**$77 642,13**	**$101 687,28**	**$199 230,28**	**$204 552,84**	**$208 261,60**	**$243 889,24**	**$276 226,85**	**$363 112,76**	**$825 062,63**	**$926 848,41**	**$1 751 911,04**

Generated by FastReport .NET

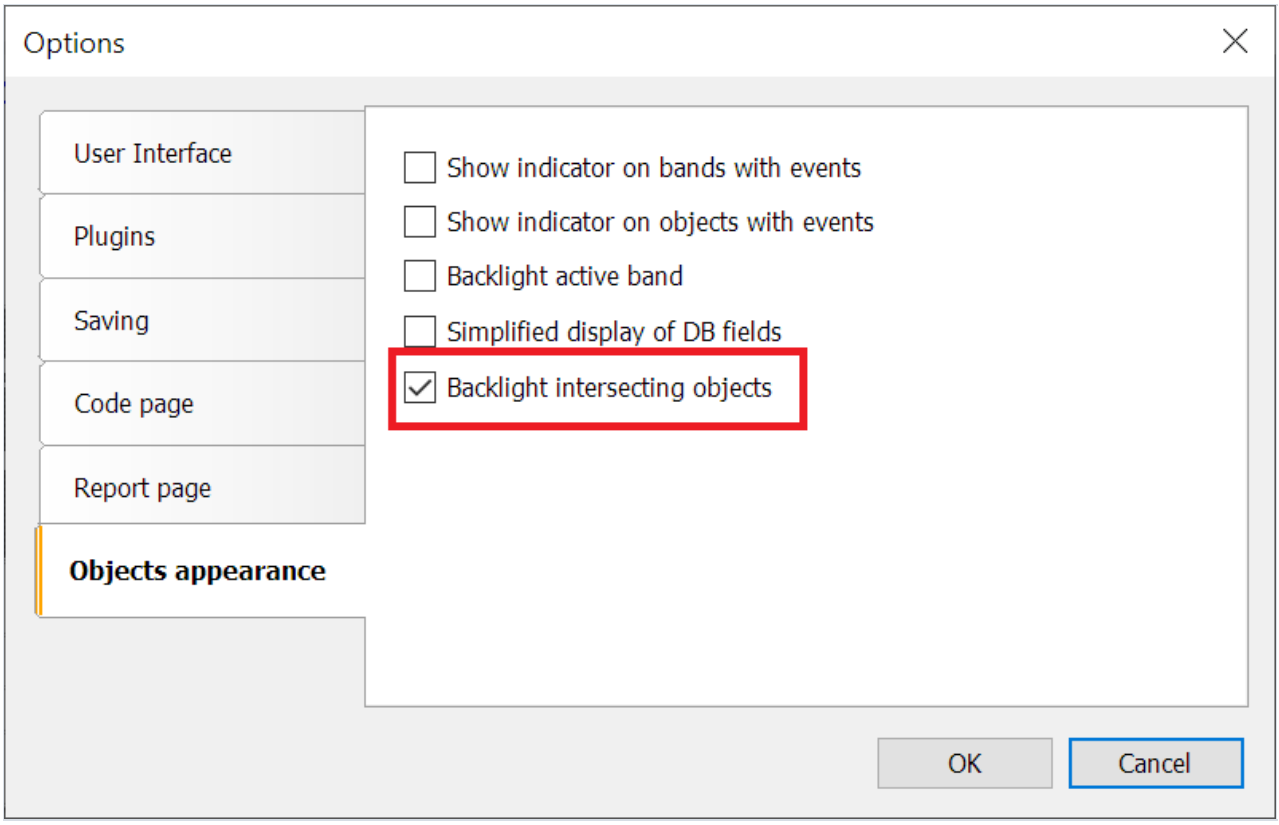
Возможность сохранить отчет со случайными данными

В меню файл в дизайнере отчетов появился новый пункт "Сохранить со случайными данными...". При таком сохранении отчета все источники данных будут сохранены в отчете и данные в них будут заменены на случайные.

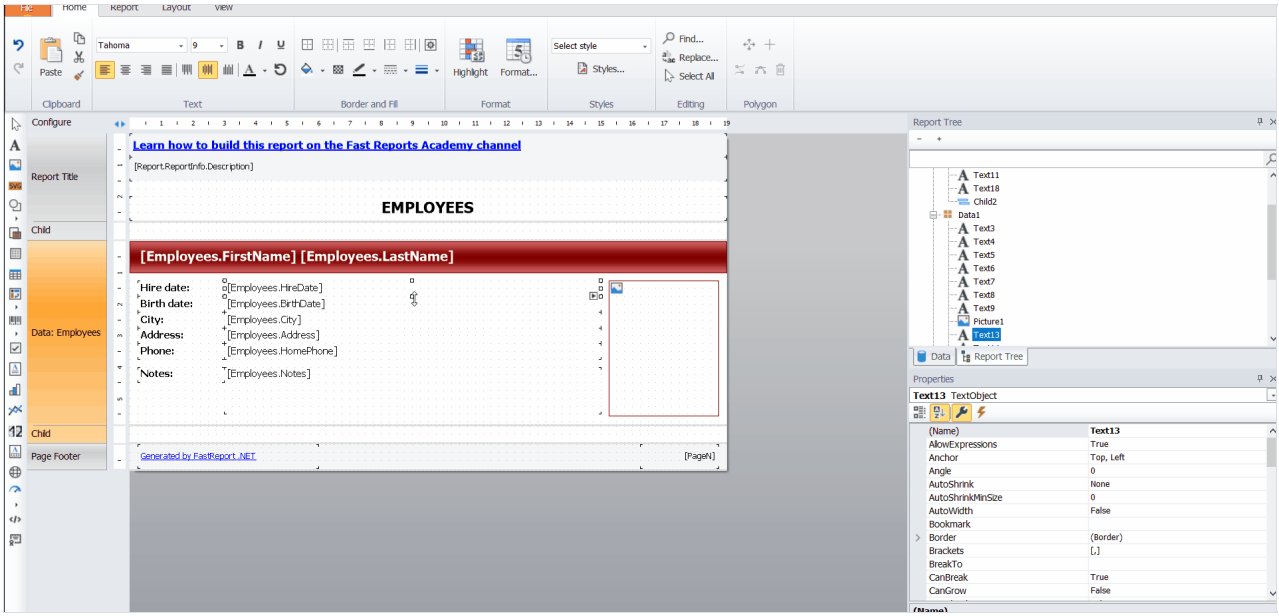
Подсветка пересекающихся объектов в дизайнере

Теперь, при размещении объектов на странице отчета, выделяются цветом пересекающиеся объекты и объекты, выходящие за пределы бэндов и страницы. В корректном отчете не должно быть таких объектов. Несоблюдение этой рекомендации, может приводить к ряду проблем при подготовке и экспорте отчетов. По умолчанию эта опция отключена.

Включить ее можно в настройках дизайнера Файл->Настройки (File->Options).



Демонстрация работы подсветки ниже:



Линейка с направляющими в редакторе RichObject

Новый инструмент позволяет удобно настраивать отступы и позиции табуляции при редактировании RichObject.

Подробнее читайте в статье.

Добавлена консольная утилита конвертации RTF

С её помощью можно удобно конвертировать RTF-файлы в шаблоны отчета.

Возможность использовать файлы XLSX в качестве источников данных.

Теперь из файлов Excel 2007 можно получить данные как из базы данных и использовать их в отчете.

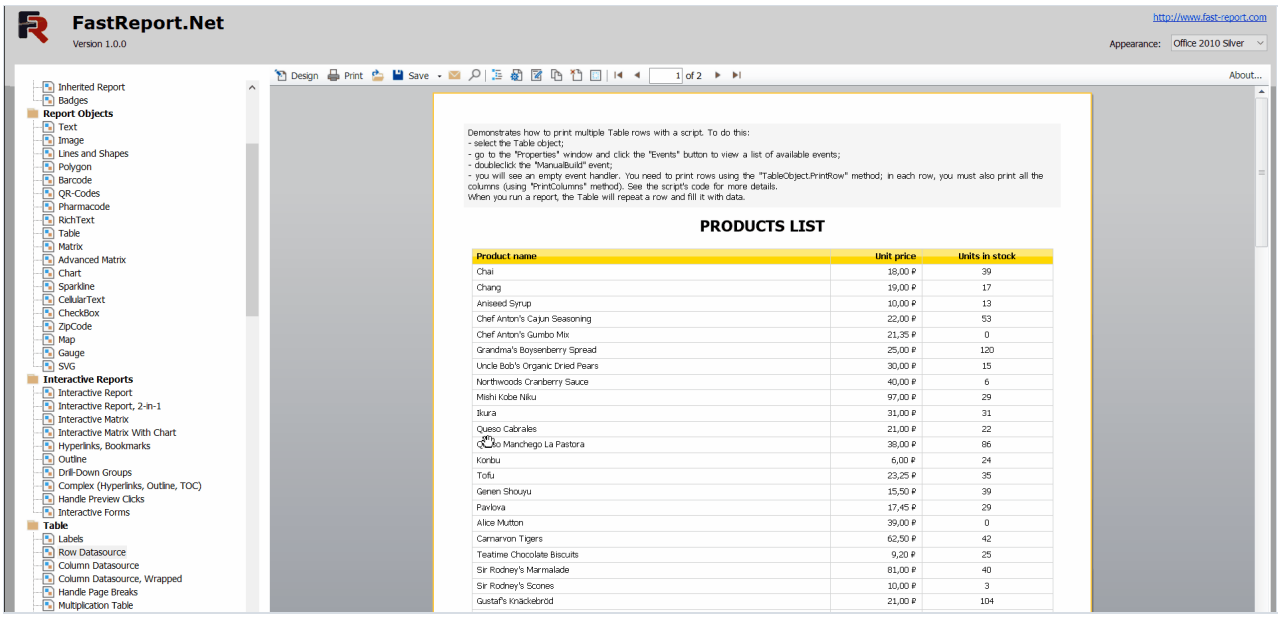
Подробнее читайте в статье.

Улучшения экспортов

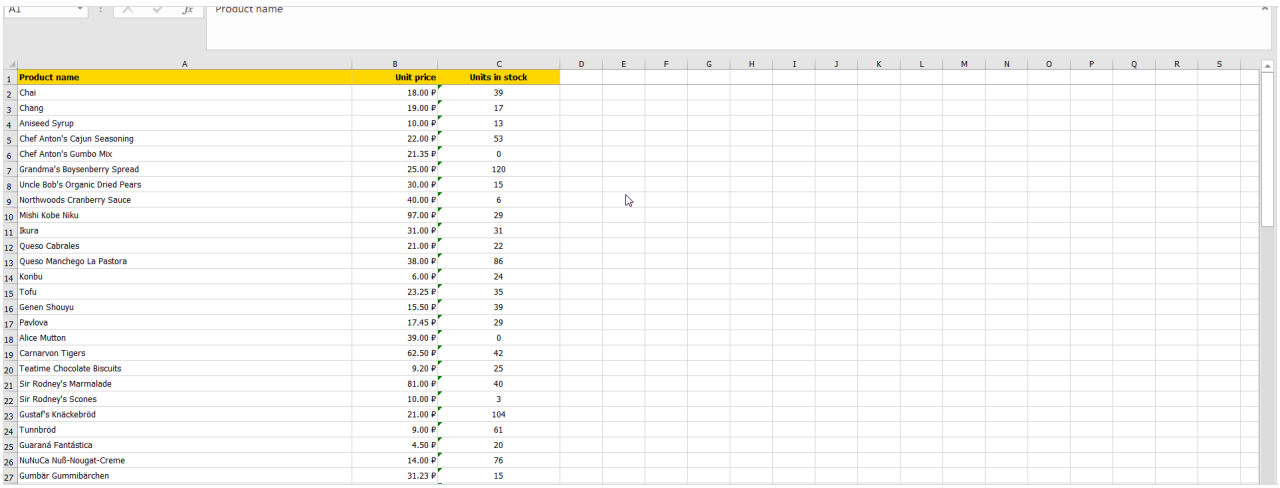
Опция "Закрепить ячейки" при экспорте в Excel 2007

Эта возможность, позволяет определить область листа, которая всегда будет видимой при прокрутке. Закрепить можно: первую строку, первый столбец, определенное количество строки и столбцов.

Настройки выглядят так:



А работает эта опция так:



Поддержка ZPL II при экспорте в ZPL

Опция "Высокое качество SVG" в HTML-экспорте

При включении этой настройки качество изображений в формате SVG будет значительно выше при экспорте. Однако, нужно учитывать, что объем занимаемой памяти так же будет выше.

Добавлен экспорт закладок и внутренних ссылок в Word

Реализован экспорт ширины символов табуляции в PDF, Word, HTML и RTF

Новое свойство PrefixStyle в SVG-экспорте

Это свойство позволяет задать префикс для всех стилей при экспорте в SVG.

Добавлен экспорт формата чисел и дат в Excel 97

Экспорт ширины символов табуляции в PDF, Word, HTML и RTF

Полный список изменений

[Engine]

- добавлена возможность сохранить отчет со случайными данными;
- в методе ExportBand теперь используется аргумент BandBase вместо Base;
- исправлены ошибки с двойным вызовом событий AfterData, BeforePrint и AfterPrint объекта ContainerObject;
- исправлена ошибка, приводящая к System.NullReferenceException при запуске отчетов с диалоговыми формами;
- исправлена ошибка, из-за которой не работало свойство VisibleExpression у подотчетов и страниц;
- исправлен баг со сдвигом по вертикали непересекающихся объектов при конвертации RTF;
- исправлена ошибка с правым якорем на страницах с бесконечной шириной и альбомной ориентацией;
- исправлена трансляция списков при конвертации RTF;
- исправлена ошибка, из-за которой не работало свойство RichObject.AllowExpressions;
- исправлена ошибка, приводящая к System.OverflowException при отрисовке неподготовленной бесконечной страницы;

[Designer]

- добавлена подсветка пересекающихся объектов;
- добавлена линейка с направляющими в редакторе RichObject;
- заменены символы пароля на точки в инспекторе объектов;
- добавлено предупреждение о возможном переполнении стека при расположении Matrix или AdvMatrix на повторяющиеся бэнды;
- удалено сообщение об ошибке, если текст штрих-кода содержит выражение;
- исправлена ошибка с опцией отключения горячих клавиш;
- исправлено выпадающее меню при нажатии на кнопку LineStyle и LineWidth;
- исправлено отображение данных в дизайнере;
- исправлены ошибки, приводящие к System.NullReferenceException при перетаскивании объектов на AdvMatrix;
- исправлена ошибка с некорректным отображением настроек тени в редакторе границ;

[Preview]

- исправлена ошибка, приводящая к System.NullReferenceException, при щелчке по редактируемому текстовому объекту;
- исправлена ошибка с неработающими гиперссылками в отчетах с много-колоночными бэндами;
- исправлена ошибка, когда экспорт отчета, приводил к сохранению подготовленного отчета;
- исправлена ошибка с настройкой списков доступных экспортов и экспортов в облака в PreviewControl;

[Exports]

- добавлен экспорт в ZPL II;
- добавлена опция "Высокое качество SVG" в HTML-экспорте;
- добавлена опция "Закрепить ячейки" при экспорте в Excel 2007;

- добавлена возможность масштабирования печати при экспорте в Excel 2007;
- добавлен экспорт закладок и внутренних ссылок в Word;
- добавлен экспорт формата чисел и дат в Excel 97;
- добавлено шифрование персональных данных пользователя в Email-экспорте;
- добавлен экспорт отступа RichObject в RTF;
- добавлен экспорт переноса строки RichObject в RTF;
- добавлен экспорт отступа TextObject в Word;
- добавлен экспорт ширины символов табуляции в PDF, Word, HTML и RTF;
- добавлено свойство PrefixStyle в SVG-экспорте, которое позволяет задать префикс для всех стилей;
- улучшен экспорт RichObject в Excel 2007;
- удалены экспорты FastReport Cloud и XMPP;
- исправлен некорректный поворот альбомной ориентации страниц при HTML-печати, если на них использовались стили с предыдущих страниц;
- исправлена ошибка с масштабом шрифта при экспорте в PDF;
- исправлена утечка памяти при экспорте SVG объектов в HTML с опцией "Высокое качество SVG";
- исправлена ошибка при встраивании шрифтов, для которых запрещена упаковка, в PDF-экспорте;
- исправлена ошибка с экспортом символов табуляции в Word;
- исправлена заливка фона изображения и свойство line-height в HTML-экспорте;
- исправлена ошибка экспорта пользовательской пунктирной линии SVGObject в PDF;
- исправлена ошибка с экспортом границ соединенных ячеек в SVG;

[WebReport]

- добавлена интерактивность расширенной матрицы в WebReport;
- исправлена обработка отмены закрытия веб-диалога в событии OnFormClosing;

[.NET Core]

- исправлена ошибка с неработающей опцией "открыть после экспорта";

[WebReport Core]

- теперь иконка DatePicker выглядит одинаково во всех браузерах;

[Demos]

- добавлено новое демо для Blazor, демонстрирующее работу с двумя отчетами;
- исправлена ошибка, из-за которой не менялся курсор при наведении на ссылки в новом демо;
- исправлена ошибка с AdvMatrix в новом демо;

[Plugins]

- добавлено подключение к Excel;
- исправлен SQLite коннектор для FastReport.Core, FastReport.CoreWin и FastReport.OpenSource;
- исправлена строка подключения к Firebird;

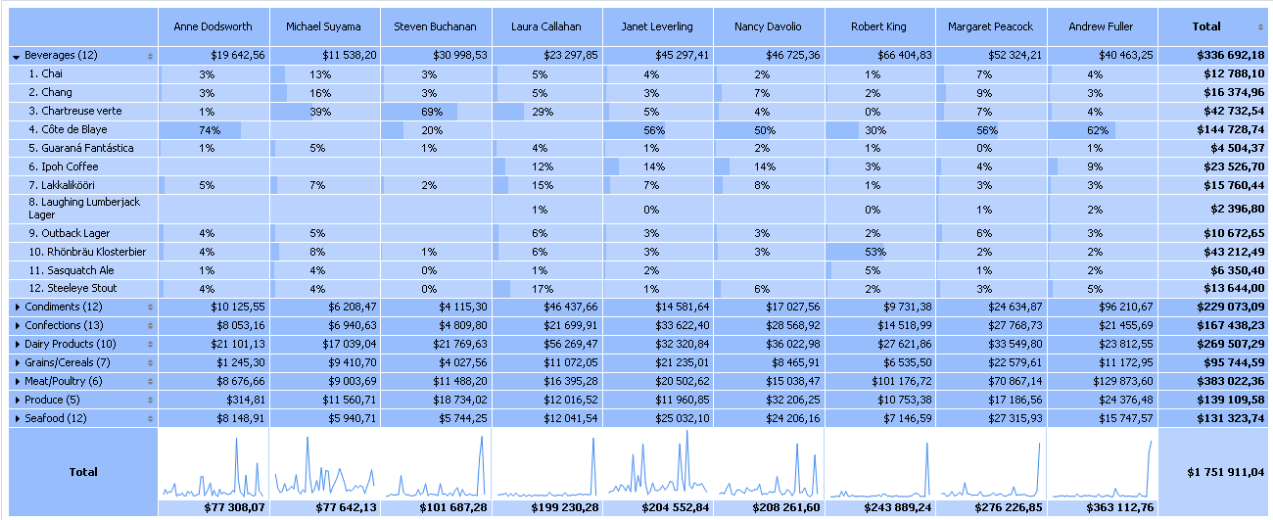
[Extras]

- добавлена утилита для конвертации RTF документов в шаблоны отчетов. (\Extras\Misc\rtf2frx).

Версия 2022.1

Новые возможности

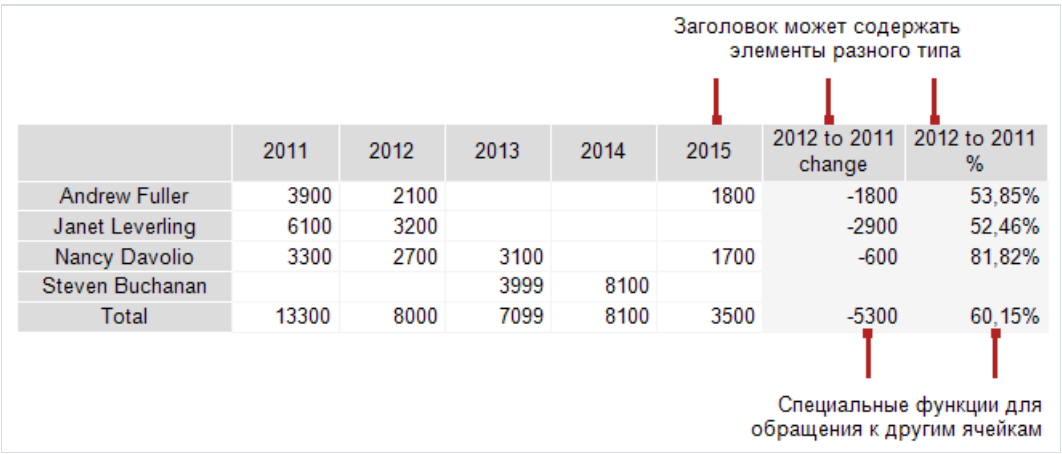
Добавлен новый объект "Улучшенная матрица":



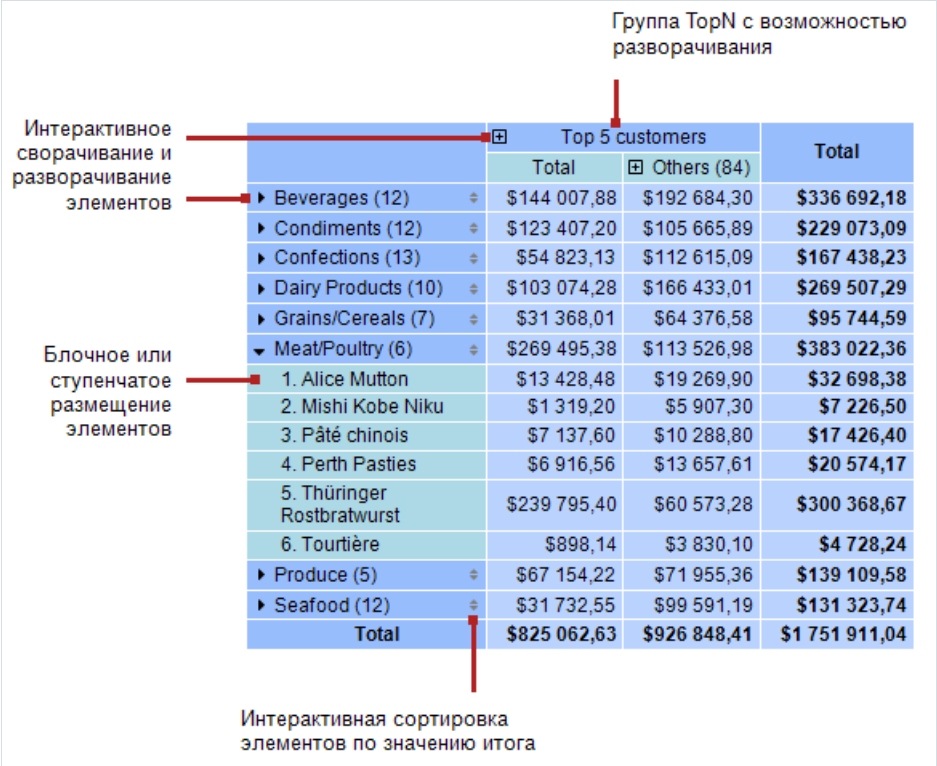
	Anne Dodsworth	Michael Suyama	Steven Buchanan	Laura Callahan	Janet Leverling	Nancy Davolio	Robert King	Margaret Peacock	Andrew Fuller	Total
Beverages (12)	$19 642,56	$11 538,20	$30 998,53	$23 297,85	$45 297,41	$46 725,36	$66 404,83	$52 324,21	$40 463,25	$336 692,18
1. Chai	3%	13%	3%	5%	4%	2%	1%	7%	4%	$12 788,10
2. Chang	3%	16%	3%	5%	3%	7%	2%	9%	3%	$16 374,96
3. Chartreuse verte	1%	39%	69%	29%	5%	4%	0%	7%	4%	$42 732,54
4. Côte de Blaye	74%		20%		56%	50%	30%	56%	62%	$144 728,74
5. Guaraná Fantástica	1%	5%	1%	4%	1%	2%	1%	0%	1%	$4 504,37
6. Ipoh Coffee				12%	14%	14%	3%	4%	9%	$23 526,70
7. Lakkalikööri	5%	7%	2%	15%	7%	8%	1%	3%	3%	$15 760,44
8. Laughing Lumberjack Lager				1%	0%		0%	1%	2%	$2 396,80
9. Outback Lager	4%	5%		6%	3%	3%	2%	6%	3%	$10 672,65
10. Rhönbräu Klosterbier	4%	8%	1%	6%	3%	3%	53%	2%	2%	$43 212,49
11. Sasquatch Ale	1%	4%	0%	1%	2%		5%	1%	2%	$6 350,40
12. Steeleye Stout	4%	4%	0%	17%	1%	6%	2%	3%	5%	$13 644,00
Condiments (12)	$10 125,55	$6 208,47	$4 115,30	$46 437,66	$14 581,64	$17 027,56	$9 731,38	$24 634,87	$96 210,67	$229 073,09
Confections (13)	$8 053,16	$6 940,63	$4 809,80	$21 699,91	$33 622,40	$28 568,92	$14 518,99	$27 768,73	$21 455,69	$167 438,23
Dairy Products (10)	$21 101,13	$17 039,04	$21 769,63	$56 269,47	$32 320,84	$36 022,98	$27 621,86	$33 549,80	$23 812,55	$269 507,29
Grains/Cereals (7)	$1 245,30	$9 410,70	$4 027,56	$11 072,05	$21 235,01	$8 465,91	$6 535,50	$22 579,61	$11 172,95	$95 744,59
Meat/Poultry (6)	$8 676,66	$9 003,69	$11 488,20	$16 395,28	$20 502,62	$15 038,47	$101 176,72	$70 867,14	$129 873,60	$383 022,36
Produce (5)	$314,81	$11 560,71	$18 734,02	$12 016,52	$11 960,85	$32 206,25	$10 753,38	$17 186,56	$24 376,48	$139 109,58
Seafood (12)	$8 148,91	$5 940,71	$5 744,25	$12 041,54	$25 032,10	$24 206,16	$7 146,59	$27 315,93	$15 747,57	$131 323,74
Total	**$77 308,07**	**$77 642,13**	**$101 687,28**	**$199 230,28**	**$204 552,84**	**$208 261,60**	**$243 889,24**	**$276 226,85**	**$363 112,76**	**$1 751 911,04**

Вот список его ключевых особенностей:

- заголовки строк и колонок могут содержать группы и простые элементы в произвольном порядке. Это позволяет строить асимметричные отчеты;
- кнопки сворачивания позволяют интерактивно управлять видимостью отдельных элементов;
- кнопки сортировки позволяют интерактивно сортировать матрицу по выбранным значениям, в том числе по значениям итогов;
- группировка Top N позволяет отобразить N значений в заголовке, а остальные значения сгруппировать в отдельный элемент с возможностью разворачивания;
- вывод заголовков матрицы в ступенчатом виде;
- сортировка заголовков по значениям итогов;
- широкий набор агрегатных функций;
- поддержка пользовательских агрегатных функций;
- широкий набор специальных функций, позволяющих получить значения итогов, соседних ячеек, а также функции для расчета процентов;
- поддержка объектов "Искрографик" и "Индикатор прогресса" в ячейках данных.



	2011	2012	2013	2014	2015	2012 to 2011 change	2012 to 2011 %
Andrew Fuller	3900	2100			1800	-1800	53,85%
Janet Leverling	6100	3200				-2900	52,46%
Nancy Davolio	3300	2700	3100		1700	-600	81,82%
Steven Buchanan			3999	8100			
Total	13300	8000	7099	8100	3500	-5300	60,15%



	Top 5 customers		Total
	Total	⊞ Others (84)	
▸ Beverages (12)	$144 007,88	$192 684,30	**$336 692,18**
▸ Condiments (12)	$123 407,20	$105 665,89	**$229 073,09**
▸ Confections (13)	$54 823,13	$112 615,09	**$167 438,23**
▸ Dairy Products (10)	$103 074,28	$166 433,01	**$269 507,29**
▸ Grains/Cereals (7)	$31 368,01	$64 376,58	**$95 744,59**
▾ Meat/Poultry (6)	$269 495,38	$113 526,98	**$383 022,36**
1. Alice Mutton	$13 428,48	$19 269,90	**$32 698,38**
2. Mishi Kobe Niku	$1 319,20	$5 907,30	**$7 226,50**
3. Pâté chinois	$7 137,60	$10 288,80	**$17 426,40**
4. Perth Pasties	$6 916,56	$13 657,61	**$20 574,17**
5. Thüringer Rostbratwurst	$239 795,40	$60 573,28	**$300 368,67**
6. Tourtière	$898,14	$3 830,10	**$4 728,24**
▸ Produce (5)	$67 154,22	$71 955,36	**$139 109,58**
▸ Seafood (12)	$31 732,55	$99 591,19	**$131 323,74**
Total	**$825 062,63**	**$926 848,41**	**$1 751 911,04**

Подробнее о возможностях объекта смотрите в документации.

Добавлены штрих-коды GS1 DataBar: Limited, Omnidirectional, Stacked и Stacked Omnidirectional.

Новые свойства: Config.CompilerSetting.ExceptionBehaviour и Config.CompilerSetting.Placeholder.

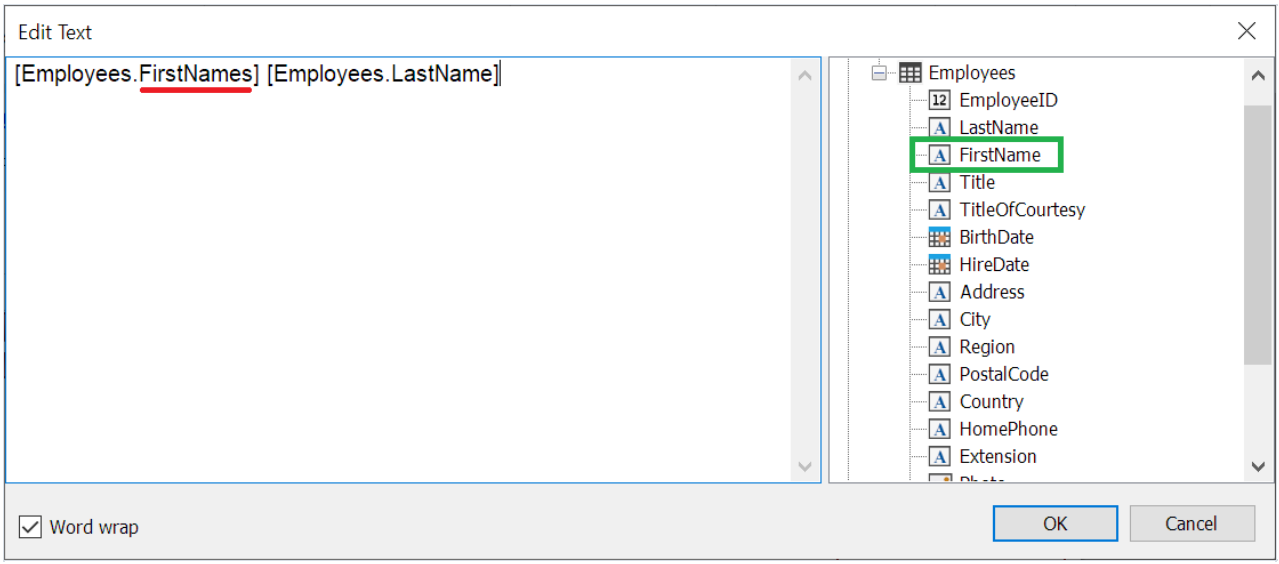
Эти свойства дают возможность настраивать поведение при возникновении исключений с некорректными именами полей и таблиц баз данных.

Config.CompilerSetting.Placeholder - строковая переменная, которая используется для замены выражений с несуществующими именами. По умолчанию, значение этой переменной - пустая строка.

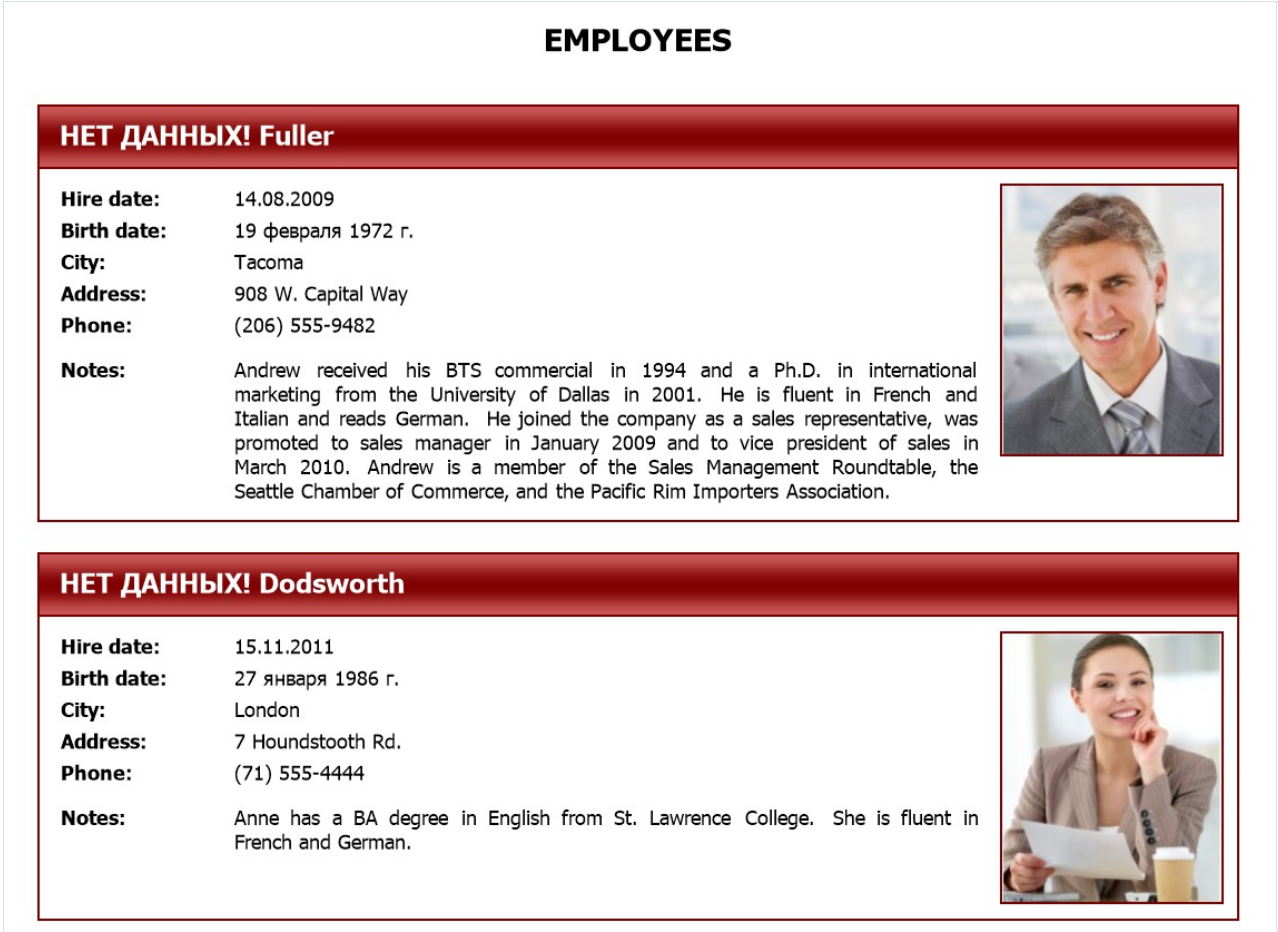
Config.CompilerSetting.ExceptionBehaviour может иметь следующие значения:
ExceptionBehaviour.Default - стандартное поведение, как и было ранее. При наличии ошибок с некорректными именами, выводится сообщение об ошибке. Подготовка отчета прерывается.
ExceptionBehaviour.ReplaceExpressionWithExceptionMessage - некорректные выражения заменяются на текст сообщения об исключении. Ошибки при этом не выводятся. Подготовка отчета не прерывается.
ExceptionBehaviour.ShowExceptionMessage - появляется сообщение с текстом исключения, после нажатия кнопки **OK**, подготовка отчета продолжается. При этом, неправильные выражения заменяются на значение переменной **Placeholder**. **ExceptionBehaviour.ReplaceExpressionWithPlaceholder** -

некорректные выражения просто заменяются на **Placeholder**. Сообщений об ошибках нет. Подготовка отчета не прерывается.

Пример при значениях переменных: **ExceptionBehaviour = ExceptionBehaviour.ReplaceExpressionWithPlaceholder Placeholder = "НЕТ ДАННЫХ!"**



Здесь видно, что в таблице есть поле с именем **FistName**, однако в выражении оно указано некорректно.



А это результат подготовки такого отчета. Ранее его подготовить было бы невозможно из-за ошибок.

Повышено качество трансляции RTF в объекты отчета.

Преобразование RTF в объекты отчета оптимизировано. Добавлена трансляция RTF в ячейках таблиц. А также исправлено много ошибок.

Улучшения экспортов

Реализован экспорт водяных знаков в Word и RTF.

Добавлено масштабирование SVG изображений в матрице экспорта.

Это позволяет повысить качество экспортируемых изображений при экспорте в Word и Excel. Однако, при этом размер выходного файла увеличивается. Для использования этой возможности, необходимо включить опцию "Для печати" при экспорте.

Реализован экспорт групп на отдельные листы в Excel 2007.

В Excel 2007 добавлена возможность экспорта свойства, определяющего размер и расположение изображения при экспорте.

Теперь можно определить как будет вести себя изображение в ячейке при изменении ее положения и размера. При этом изображение может:

- перемещаться и изменять свой размер вместе с ячейкой
- перемещаться вместе с ячейкой, но не изменять свой размер
- не перемещаться и не изменять размер

Реализована возможность скрывать или показывать линии сетки при экспорте в Excel 97.

В HTML-экспорте добавлена опция "Не поворачивать альбомные страницы при печати".

Ранее мы принудительно поворачивали страницы с альбомной ориентацией при печати. Необходимо это было по причине того, что браузеры не умеют корректно печатать отчеты со страницами как в книжной, так и в альбомной ориентации. При печати таких документов, страницы с альбомной ориентацией обрезаются по ширине страниц с книжной ориентацией. Теперь, можно настроить, поворачивать страницы с альбомной ориентацией или нет. Кроме того, исправлена ошибка, при которой страницы с альбомной ориентацией поворачивались всегда, даже когда нет страниц с книжной ориентацией.

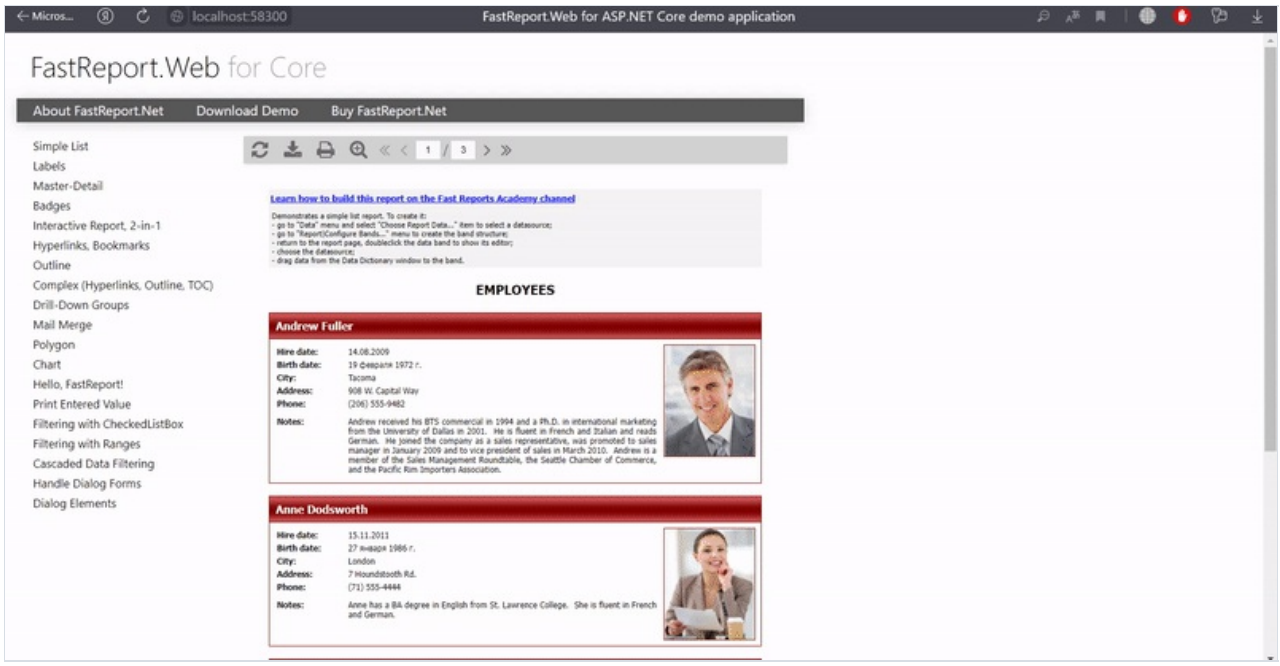
Поддержка .NET 6

Добавлена поддержка .NET 6 для FastReport.Core и FastReport.CoreWin.

Улучшения WebReport for Core и Blazor Server

Настройки экспортов

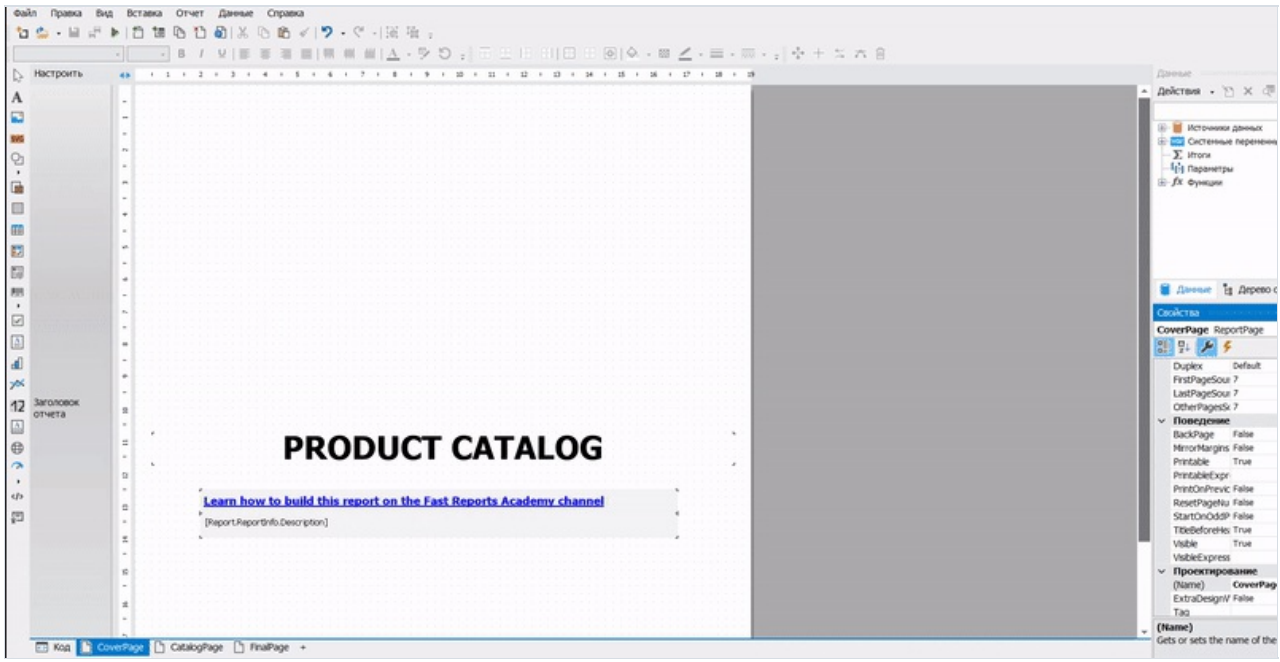
Добавлены возможности по настройке экспорта отчёта в необходимый формат из WebReport. При активации свойства **webReport.Toolbar.Exports.EnableSettings** рядом с кнопкой необходимого экспорта появится иконка настройки. При нажатии на эту кнопку отобразится окно со списком возможных настроек (экспортируемые страницы, свойства и др.).



Окно настроек экспорта можно кастомизировать, используя свойства **webReport.Toolbar.Exports.Color** и **webReport.Toolbar.Exports.FontSettings**.

Разделение разных страниц отчёта в закладки.

Добавлена возможность открыть разные страницы отчёта (ReportPage) в разных вкладках WebReport. Для активации этой возможности, необходимо включить опцию **webReport.SplitReportPagesInTabs**.



Статические стили в WebReport.

Для возможности переопределить стандартные стили тулбара, outline и других элементов для собственной кастомизации были добавлены статические имена классов. К ним относятся: `fr-toolbar`, `fr-toolbar-item`, `fr-toolbar-narrow`, `fr-toolbar-dropdown-content`, `fr-toolbar-zoom-selected`, `fr-toolbar-pointer`, `fr-toolbar-notbutton`, `fr-toolbar-slash`.

Полный список изменений

[Engine]

- добавлен новый объект AdvMatrixObject

- добавлены штрих-коды GS1 Databar: Limited, Omnidirectional, Stacked и Stacked Omnidirectional
- добавлены новые свойства: Config.CompilerSetting.ExceptionBehaviour и Config.CompilerSetting.Placeholder. Эти свойства дают возможность настраивать поведение при возникновении исключений с некорректными именами полей и таблиц баз данных.
- добавлена трансляция RichObject внутри TableCell
- переработана трансляция RichObject в объекты отчёта
- исправлен ShiftMode транслированных из RTF текста объектов
- исправлена ошибка с двумя параметрами с одним именем приводящая к исключению System.ArgumentException
- исправлена ошибка с вложенным отчетом содержащим многоколоночный Databand
- исправлена ошибка с неверным расчетом высоты бэнда
- исправлена ошибка с отображением гиперссылок при конвертации RTF в объекты отчета
- исправлена трансляция RichObject если документ назначается из скрипта
- исправлена ошибка с шрифтами, добавляемыми в Config.PrivateFontCollection

[.Net Core]

- добавлена поддержка .NET 6
- исправлен некорректный поиск Bold-Italic шрифтов

[Designer]

- добавлена проверка вводимых данных в окне редактирования QR-кода СберБанка
- исправлена ошибка с переносом строки в редакторе текстового объекта
- исправлена ошибка конвертации отчетов rdl, содержащих матрицы внутри ячеек таблицы
- исправлена ошибка с направляющими линиями в дизайнере
- исправлена ошибка с окном "Дерево отчета"
- исправлена ошибка приводящая к System.NullReferenceException и падению дизайнера во время его запуска при включенной опции Авто-направляющие

[Preview]

- исправлена ошибка со сдвигом позиции объектов при переключении вида бэндов во время редактировании подготовленной страницы

[Exports]

- реализован экспорт водяных знаков в Word
- реализован экспорт водяных знаков в RTF
- добавлена опция "Не поворачивать альбомные страницы при печати" при экспорте в HTML
- добавлена возможность изменять имя прикрепляемого файла при отправке по Email
- добавлено масштабирование SVG изображений в матрице экспорта
- добавлена возможность экспорта свойства, определяющего размер и расположение изображения при экспорте в Excel 2007
- реализована возможность скрывать или показывать линии сетки при экспорте в Excel 97
- реализован экспорт групп на отдельные листы в Excel
- реализован экспорт уровня прозрачности изображений водяных знаков в Word
- реализован экспорт размера изображения водяного знака в RTF
- исправлена ошибка, приводившая к System.NullReferenceException при экспорте в текст, таблиц с количеством строк меньше одной
- исправлен некорректный левый отступ таблиц при экспорте в Word
- исправлена ошибка со шрифтом Wingdings в тегах HTML при экспорте в HTML
- исправлен баг с экспортом шрифтов Wingdings и Webdings в HTML
- исправлена ошибка с шириной фреймов при экспорте в PowerPoint

- исправлена ошибка экспорта объектов с прозрачной заливкой в RTF
- исправлена ошибка экспорта объектов с прозрачной заливкой в Word
- исправлена ошибка, приводящая к System.OutOfMemoryException при экспорте в PDF
- исправлено неправильное отображение переноса строки при экспорте в HTML
- исправлено переполнение памяти при экспорте в PDF
- исправлены ошибки в PDF-экспорте на системах, отличных от Windows
- исправлена ошибка с экспортом таблиц с количеством столбцов больше 63 в Word 2007
- исправлена ошибка приводящая к утечке памяти и исключению System.OutOfMemoryException в PDF-экспорте при включенной опции "Текст в кривых"
- исправлена ошибка с разрывом строки при экспорте в HTML

[WebReport]

- исправлена ошибка с символом новой строки при использовании шрифта Wingdings

[WebReport Core/Blazor Server]

- добавлены возможности настраивать свойства экспорта отчёта из WebReport. При активации свойства webReport.Toolbar.Exports.EnableSettings рядом с кнопкой необходимого экспорта появится иконка настройки
- добавлено свойство webReport.SplitReportPagesInTabs, которое позволяет разделить разные страницы отчёта (ReportPage) в разных вкладках WebReport
- добавлены статические имена классов для возможности переопределить стандартные стили тулбара, outline и других элементов
- исправлено обновление WebReport при вводе значения с клавиатуры в поле DateTimePicker
- исправлена ширина табов (tabs) при нестандартных размерах страницы отчёта

[Demos]

- добавлена демо использования WebReport Core для .NET 5
- добавлена демо использования WebReport Core для Angular
- добавлена демо использования WebReport Blazor для Blazor Server

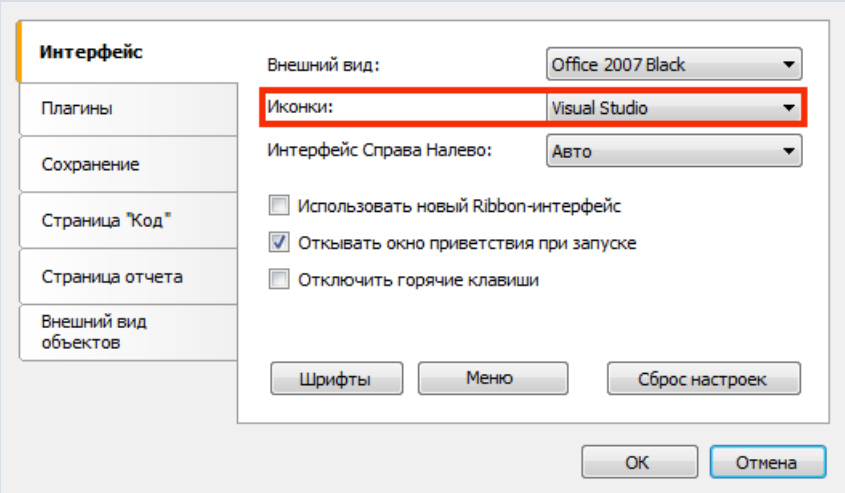
Версия 2021.4

Изменения редакций

Редакций WinForms и Win+WebForms теперь нет. Вместо них добавлена новая редакция Standard, включающая Windows Forms компоненты, ASP.NET компоненты, а также поддержку .NET Core / .NET 5 / Blazor.

Новые возможности

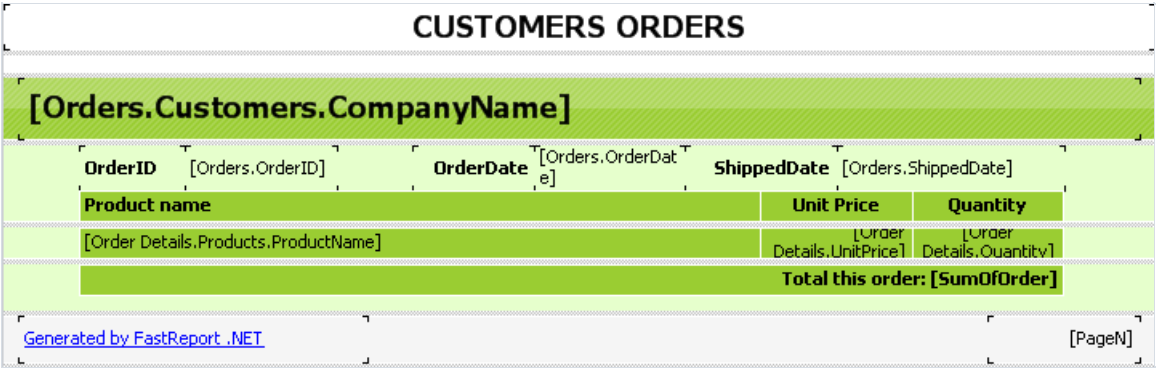
- Добавлены новые иконки в стиле Visual Studio. Вы можете переключаться между наборами иконок в окне дизайнера "Вид/Настройки/Интерфейс" (или "Файл/Настройки/Интерфейс" в случае использования ribbon-интерфейса):



  Новые иконки представлены в нескольких размерах и отлично подходят для работы на мониторах высокого разрешения (hiDPI).

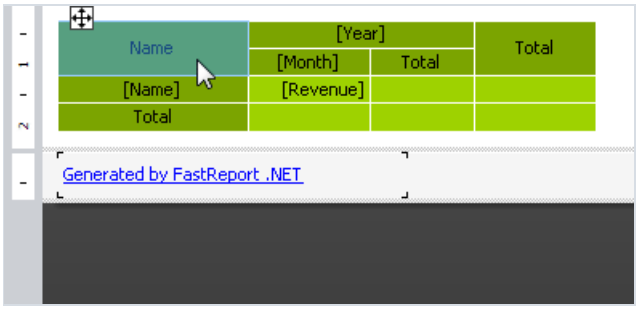
- Добавлена возможность сброса настроек среды FastReport, которые хранятся в конфигурационном файле `FastReport.config`. Это можно сделать в окне дизайнера "Вид/Настройки/Интерфейс", кнопка "Сброс настроек". Для вступления изменений в силу требуется перезапуск дизайнера.

- Добавлена возможность упрощенного отображения полей БД в объекте "Текст" в режиме дизайна. Эта настройка доступна в окне дизайнера "Вид/Настройки/Внешний вид объектов". По умолчанию настройка отключена; при ее включении объекты "Текст", содержащие одно поле БД, отображаются в упрощенном виде:



  Это улучшает визуальное восприятие отчета, имеющего много мелких полей. Полное содержимое поля по-прежнему отображается в строке статуса и при редактировании текста объекта.

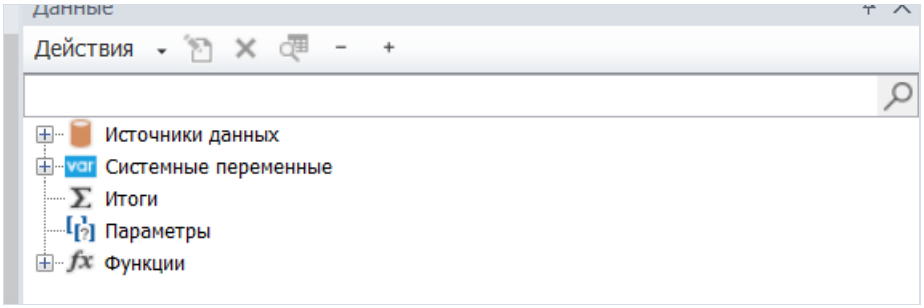
- Добавлена возможность индивидуальной настройки ячеек в углу объекта "Матрица". Для этого используйте контекстное меню ячейки, команды "Разбить ячейку", "Объединить ячейки":



- Добавлена возможность подключения к ElasticSearch. Подключение доступно в мастере подключения к данным и из кода.

- Добавлен штрих-код Japanese Post 4 - State Code.

- Добавлены кнопки свернуть все/развернуть все, и поле поиска для дерева отчетов и дерева данных в дизайнере. При нажатии на + дерево будет развернуто. На - дерево будет свернуто.



  Эти изменения должны упростить работу с отчетами, содержащими много объектов и/или источников данных.

- Значительно улучшен и оптимизирован конвертор RicthText в объекты отчета.

- Количество доступных экспортов в WebReport Core/Blazor Server значительно увеличено.

- Добавлены объекты интеграции с FastReport Business Graphics (\Extras\Objects\FastReportBGObjects).

Универсальные плагины FastReport.Data

Обновлены пакеты с плагинами-коннекторами FastReport.Data. Теперь они включают в себя плагины для разных редакций FastReport (.NET, Core, CoreWin, OpenSource) и автоматически подключают нужную библиотеку, в зависимости от используемого продукта. Для корректной работы необходима редакция FastReport версии 2021.3.0 и выше.

Плагины-коннекторы FastReport.Core.Data, FastReport.CoreWin.Data, FastReport.OpenSource.Data объявлены как устаревшие и более не поддерживаются.

Улучшения в публикации пользовательских приложений с использованием FastReport

Для пользовательских приложений на .NET Core 3.0+ и .NET 5+, использующих FastReport.Core, FastReport.CoreWin, FastReport.OpenSource была добавлена поддержка Single File Applications (SFA).

Также, добавлена поддержка публикации приложения с обрезанием неиспользуемых библиотек - свойство MSBuild - PublishTrimmed*.

Внимание! В некоторых случаях, возможно, придётся явно указывать список сборок, которые .NET не должен обрезать. Это может понадобится, если в скрипте отчёта используются эти библиотеки, однако в коде вашего приложения они не задействуются.

Делается это с помощью свойства TrimmerRootAssembly. В данном случае, например, явно указывается, что библиотеку System.Security обрезать не нужно:

```
<ItemGroup>
  <TrimmerRootAssembly Include="System.Security" />
</ItemGroup>
```

Локализации

В логике смены локализации были сделаны небольшие изменения.

1. Добавлен пакет **FastReport.Localization**. Данный пакет содержит файлы локализации для продуктов FastReport.NET, FastReport.Core, FastReport.CoreWin, FastReport.Mono, FastReport.OpenSource и при добавлении этого пакета создает директорию Localization в выходной директории пользовательского проекта.

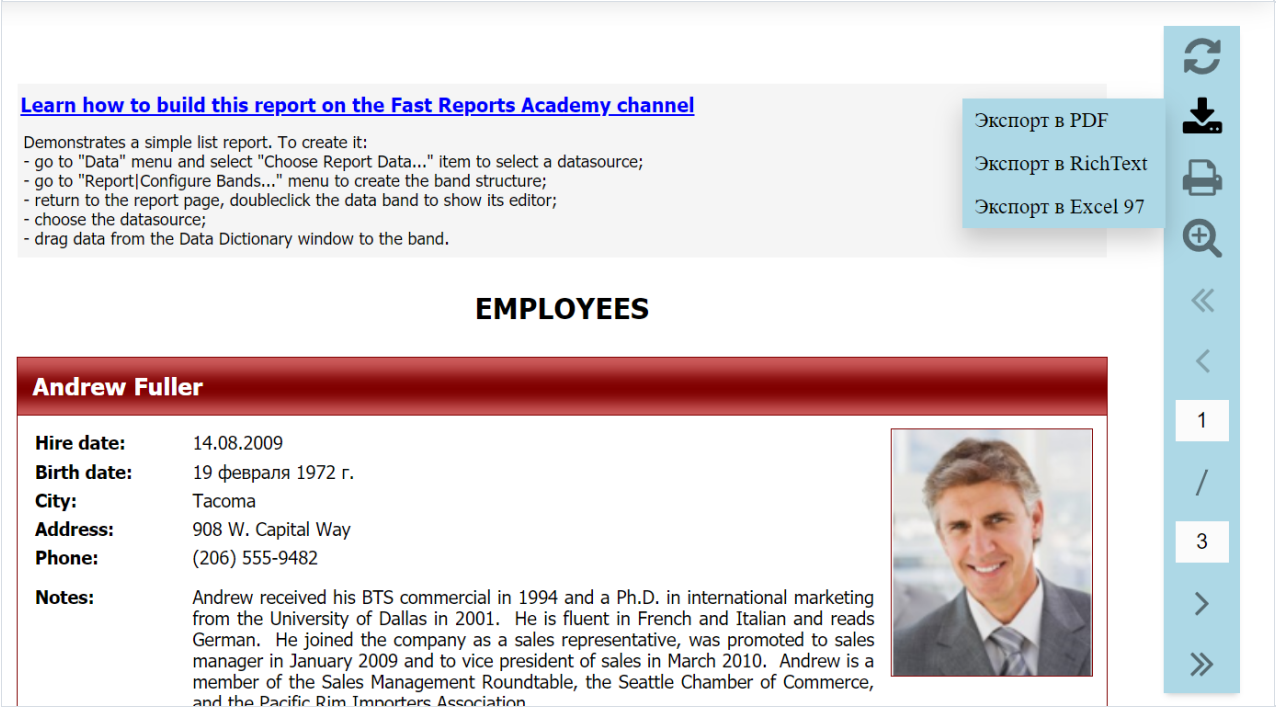
2. Добавлен новый API для смены локализации используя тип CultureInfo - *FastReport.Utils.Res.LoadLocale(CultureInfo culture)*.

   При вызове данного метода, FastReport ищет подходящую локализацию для выбранной культуры. Загруженные локализации кэшируются. Для корректной работы этого метода, необходимо установить в свой проект пакет FastReport.Localization из п.1 или установить путь к папке с файлами локализации в свойстве *FastReport.Utils.Res.LocaleFolder*.

Изменения и улучшения в тулбаре WebReport Core/Blazor

- Настройки тулбара были вынесены из класса WebReport в свойство WebReport.Toolbar класса ToolbarSettings.
- Добавлена возможность настройки тулбара: Положение, цвет выпадающего меню, шрифт, прозрачность иконок, изменение цвета иконок, изменение положения контента. Данные свойства доступны в webReport.Toolbar
- Во время загрузки отчёта тулбар теперь не отображается.
- У объекта Toolbar было добавлено свойство ShowOnDialogPage (по умолчанию true), которое позволит отключить отрисовку тулбара, если в данный момент открыто диалоговое окно
- Добавлено больше экспортов в выпадающем меню тулбара. Данные свойства доступны в webReport.Toolbar.Exports.ExportTypes. Список добавленных экспортов: HTML, Hpgl, Dxf, Json, LaTeX, Ppml, PS, Xaml, Zpl, Excel97, Svg.

```
ToolbarSettings toolbar = new ToolbarSettings()
{
    Color = Color.LightBlue,
    DropDownMenuColor = Color.LightBlue,
    ShowOnDialogPage = false,
    DropDownMenuTextColor = Color.Black,
    IconColor = IconColors.Black,
    Position = Positions.Right,
    FontSettings = new Font("Arial", 14, FontStyle.Bold),
    Exports = new ExportMenuSettings()
    {
        ExportTypes = Exports.Pdf | Exports.Excel97 | Exports.Rtf
    }
    // or
    //Exports = ExportMenuSettings.All
};
webReport.Toolbar = toolbar;
```

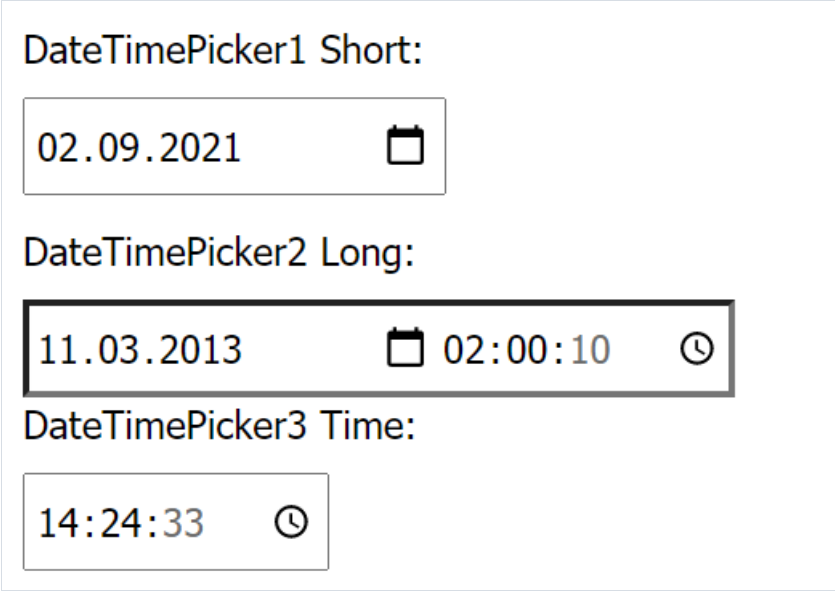


Улучшения WebReport Core/Blazor Server

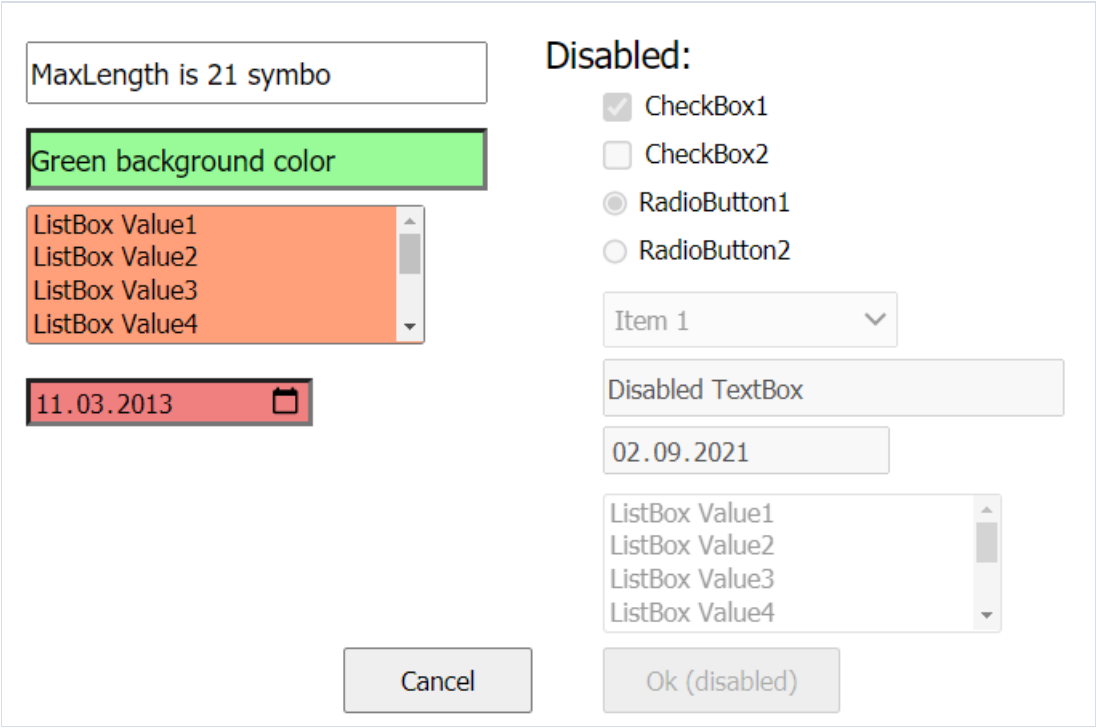
Добавлена поддержка Blazor Server компонентов для пакета FastReport.Core3.Web (CoreWin).

Улучшена поддержка компонентов диалоговых отчетов:

- DateTimePicker был улучшен. В режиме DateTimePicker.Format.Time он отображает только время, в режиме DateTimePicker.Format.Short - только дату, DateTimePicker.Format.Long - и дату и время



- Добавлена поддержка свойства MaxLength для TextBox
- Добавлена поддержка свойства Enabled
- Добавлена поддержка background-color



Исправления

- Исправлена ошибка, связанная с работой свойств `Dock` и `Anchor` у объектов, которые находятся внутри ячеек таблицы или матрицы.
- Исправлена ошибка, приводящая к переполнению стека, при размещении объекта "Вложенный отчет" на подвале страницы.
- Исправлена ошибка при экспорте в формат SVG, если в системе используется нестандартная настройка DPI.

Полный список изменений

[Engine]

- добавлено подключение к ElasticSearch
- добавлен новый штрих-код - Japanese PostNet

- добавлен метод Res.LoadLocale(CultureInfo), который изменяет выбранную локаль по аргументу CultureInfo. Загруженные локали кэшируются. Для корректной работы требуется добавленный пакет FastReport.Localization
- оптимизирован и унифицирован конвертор RichText в объекты отчёта
- исправлена ошибка с неправильной шириной табуляции при TextObject.TextRenderType = TextRenderType.HtmlTextRenderer
- исправлена ошибка с объектом "Вложенный отчет" на бэнде "Подвал страницы", приводящая к переполнению стека
- исправлена ошибка со свойствами Dock и Anchor у объектов внутри ячеек таблицы/матрицы
- исправлена ошибка, приводящая к System.ArgumentException при отрисовке PictureObject расположенными за пределами бэнда
- исправлена ошибка с некорректной работой правого якоря (Anchor = AnchorStyles.Right) при неограниченной ширине страницы
- исправлена ошибка с заменой пользовательского шрифта на шрифт, по умолчанию при подготовке отчета
- исправлена ошибка с выравниванием по вертикали при конвертации RTF (по умолчанию теперь Top вместо Center).
- исправлена ошибка конвертации таблиц RTF в объекты отчета

[Designer]

- добавлено упрощенное отображение имен полей БД в дизайнере
- добавлена кнопка свернуть все/развернуть все, и поле поиска для дерева отчетов и дерева данных
- добавлены новые иконки. Вы можете выбрать их в окне дизайнера "Вид/Настройки/Интерфейс".
- исправлена ошибка, приводящая к падению дизайнера отчётов, при некорректной таблице в источнике данных

[Preview]

- исправлена ошибка сохранения подготовленных отчётов, содержащих сконвертированный RichObject

[Exports]

- добавлена опция при экспорте в Word 2007 "Не добавлять разрывы разделов при разрывах страниц". По умолчанию добавляются и разрывы страниц, и разрывы разделов.
- исправлен разрыв страницы в Html экспорте (свойство PageBreaks)
- исправлен SVG экспорт с параметром "Экспорт в несколько файлов"
- исправлена ошибка в SVG экспорте на мониторах с высоким разрешением
- исправлены имена фалов, сохраняемых в zip архив
- исправлена ширина табуляции при экспорте RichObject
- исправлена ошибка XPS-экспорта, при которой документы, экспортированные на Linux, не открывались на Windows
- исправлены ошибки некорректной работы свойств Anchor и Dock при экспорте страниц с бесконечной шириной
- исправлена ошибка экспорта текстовых объектов в Excel 2007 с включенным типом рендеринга HtmlParagraph. Отключите параметр экспорта WYSIWYG, чтобы экспортировать текст вместо изображений.

[WebReport]

- для диалогов в WebReport добавлена поддержка background-color
- для диалогов в WebReport добавлена поддержка свойства Enabled
- для диалогового компонента TextBox в WebReport добавлена поддержка свойства MaxLength
- исправлен некорректный фоновый цвет страницы при HTML/Blazor экспорте на браузерах Safari

- исправлена ошибка с зависанием обратного вызова сохранения онлайн-дизайнера в WebReport с сессиями
- исправлены ошибки некорректной работы свойств Anchor и Dock на страницах с бесконечной шириной
- оптимизирована загрузка локализации для Toolbar

[Online Designer]

- исправлено сохранение/вызов превью из Онлайн-Дизайнера со страницей в Landscape ориентации

[.Net Core]

- добавлена поддержка Single File Application
- обновлены зависимости к FastReport.Compat и FastReport.DataVisualization. FastReport.Compat теперь правильно определяет возможность использования WinForms API. FastReport.DataVisualization теперь не имеет зависимость к System.Data.SqlClient и System.Drawing.Common
- исправлена ошибка, при которой отчет не работал с данными из пользовательской библиотеки, хотя она была зарегистрирована в ReferencedAssemblies в CoreWin
- исправлен вылет приложения при загрузке отчета с неизвестным шрифтом в нескольких потоках на Linux
- исправлена ошибка "Could not load type 'System.Drawing.Design.UITypeEditor'"
- исправлена загрузка имен таблиц в XmlDataConnection
- исправлена ошибка, из-за которой не загружался отчёт и ресурсы при публикации/отладки используя IIS/IIS Express. Для корректной работы, необходимо вызывать метод `UseFastReport()` перед 'UseMvc/UseEndpoints'

[WebReport Core/Blazor Server]

- добавлена поддержка Blazor компонентов для пакета FastReport.Core3.Web
- исправлен некорректный вывод мультистрочного текста в Blazor (Interactive Forms & TextBox)
- добавлены xml комментарии (DocumentationFile) к Web библиотекам
- добавлено свойство для отключения отображения тулбара на диалоговой странице отчета: webReport.Toolbar.ShowOnDialogPage
- добавлено больше экспортов в выпадающем меню тулбара. Данные свойства доступны в webReport.Toolbar.Exports.ExportTypes. Список добавленных экспортов: HTML, Hpgl, Dxf, Json, LaTeX, Ppml, PS, Xaml, Zpl, Excel97, Svg.
- добавлена возможность настройки тулбара: Положение, цвет выпадающего меню, шрифт, прозрачность иконок, изменение цвета иконок, изменение положения контента. Данные свойства доступны в webReport.Toolbar
- диалоговый DateTimePicker для WebReport был улучшен. В режиме DateTimePicker.Format.Time он отображает только время, в режиме DateTimePicker.Format.Short - только дату, DateTimePicker.Format.Long - и дату, и время.
- для диалогового компонента Label в WebReport исправлено отсутствие переносов строк

[Extras]

- добавлен пакет 'FastReport.Localization', который включает в ваш проект файлы локализации FastReport для работы с разными языками
- добавлены объекты интеграции с FastReport Business Graphics (\Extras\Objects\FastReportBGObjects)

[Demos]

- реализован переход к списку отчетов, при нажатии на стрелку на папке в новом демо
- изменен Target Framework для нового демо на 4.7.2
- изменен цвет неактивных кнопок в режиме отображения миниатюр нового демо

- изменен цвет фона ползунка масштабирования в новом демо
- изменен цвет фона при отображении диалоговых форм в новом демо
- изменен цвет фона вкладок интерактивных отчетов в новом демо
- изменено расположение папки с миниатюрами отчетов для демонстрационного приложения. Теперь эта папка располагается не в Program Files а в AppData\Local

- исправлены проблемы с отображением элементов интерфейса нового демонстрационного приложения
- исправлена ошибка, вызывающая сохранение подготовленного отчета при нажатии на раскрывающиеся элементы в меню сохранения нового демонстрационного приложения
- исправлена ошибка с выравниванием отчетов в окне предварительного просмотра нового демо
- исправлена ошибка двойного запуска диалоговых форм при выборе отчета в новом демо
- исправлена ошибка с растяжением миниатюр в новом демо
- исправлена ошибка, приводящая к задержке при перемещении окна нового демо
- исправлена ошибка нового демо с одновременным отображением миниатюр в панели папок и в панели отчета

[Plugins]

- обновлены пакеты с плагинами-коннекторами FastReport.Data. Теперь они включают в себя плагины для разных редакций FastReport (.NET, Core, CoreWin, OpenSource) и автоматически подключают нужную библиотеку, в зависимости от используемого продукта
- postgres npgsql понижение версии с 4.0.3 до 3.2.7

Контакты и техподдержка

Вы всегда можете задать вопросы по использованию продукта с помощью email, либо с помощью формы на сайте.

Также мы будем рады вашим предложениям по улучшению нашего продукта.