Версия 2021.1

Руководство программиста FastReport Mono

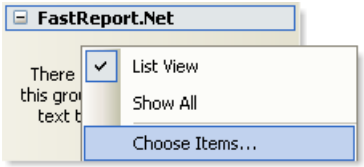
Общая информация

Установка в Toolbox

Программа установки FastReport.Net автоматически добавляет компоненты FastReport в Visual Studio Toolbox. Если по каким-то причинам этого не произошло, вы можете выполнить ручную установку компонентов.

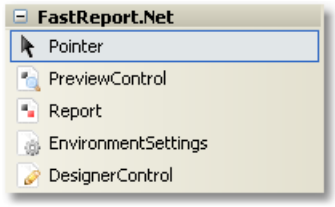
Для ручной установки компонентов FastReport.Net в Visual Studio Toolbox сделайте следующее:

- 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.Net и находится в папке:

Windows XP:

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

Windows Vista:

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

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 - основная библиотека FastReport.Net;
- FastReport.Web.dll - библиотека для работы в ASP.Net, содержит компонент WebReport;
- FastReport.Bars.dll - библиотека для организации плавающих окон, панелей инструментов и меню;
- FastReport.Editor.dll - редактор кода с подсветкой синтаксиса. Эта библиотека не нужна, если ваше приложение не использует дизайнер отчетов;
- FastReport.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.

Также вы можете распространять файл справки FRNetUserManual.chm. Этот файл может быть вызван из меню "Справка" в дизайнере отчетов.

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

Компиляция исходного кода

В комплект версии FastReport.Net Professional включены исходные коды библиотек FastReport.dll, FastReport.Web.dll. Вы можете включить их в состав своего приложения. Покажем, как сделать это:

- 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");
- добавьте файл проекта "FastReport.Web.csproj" (он находится в папке "C:\Program Files\FastReports\FastReport.Net\Source\FastReport.Web").

Отключите подписывание сборок FastReport и FastReport.Web. Для этого:

- щелкните правой кнопкой мыши на проекте "FastReport";
- выберите пункт меню "Properties";
- перейдите на закладку "Signing" и отключите флажок "Sign the assembly";
- проделайте то же самое для сборки FastReport.Web.

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".

Работа в Windows.Forms

Использование компонента Report в Visual Studio

Работа с отчетом в коде

Хранение и загрузка отчета

Регистрация данных

Передача параметра в отчет

Запуск отчета

Запуск дизайнера

Экспорт отчета

Конфигурация среды FastReport.NET

Подмена диалогов "Открыть" и "Сохранить"

Подмена стандартного окна прогресса

Передача в отчет строки подключения

Передача в отчет текста SQL

Обращение к объектам отчета

Создание отчета с помощью кода

Использование собственного окна просмотра

Фильтрация списка таблиц в "Мастере подключения"

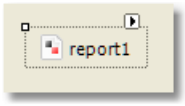
Использование компонента Report в Visual Studio

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

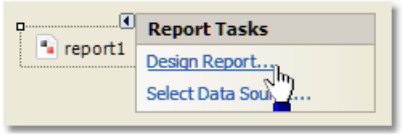
- создайте проект Windows Forms Application;
- добавьте в него источник данных (в меню "Data|Add New Data Source...");
- переключитесь на дизайнер формы;
- добавьте на форму компонент DataSet из категории "Data" и подключите его к типизированному источнику данных, который вы создали ранее.

Чтобы создать отчет, выполните следующие шаги:

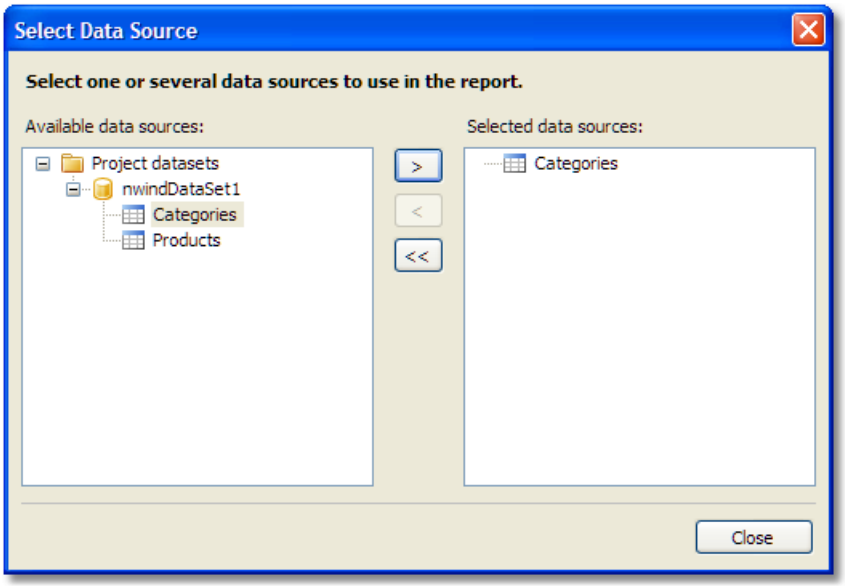
- положите на форму компонент Report:



- 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	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).	
`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 вызовуRegisterData(data, name, 1). Используя меню "Данные	Выбрать данные для отчета...", можно раскрыть структуру бизнес-объекта до нужной глубины вложенности.

МЕТОД	ОПИСАНИЕ	
`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.Net, кроме версии Basic, 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.

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.Net

С помощью компонента EnvironmentSettings, доступного в Toolbox, вы можете поменять некоторые настройки среды FastReport.Net. Для этого положите компонент на форму и измените нужные настройки в окне Properties.

Настройки, относящиеся к отчету, доступны в свойстве EnvironmentSettings.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 EnvironmentSettings.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 EnvironmentSettings.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 EnvironmentSettings.EmailSettings:

СВОЙСТВО	ОПИСАНИЕ
`string Address`	Адрес отправителя (ваш адрес).
`string Name`	Имя отправителя (ваше имя).

СВОЙСТВО	ОПИСАНИЕ
`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.Net.

Настройки стиля интерфейса доступны в следующих свойствах:

СВОЙСТВО	ОПИСАНИЕ
`UIStyle UIStyle`	Стиль интерфейса окна дизайнера и предварительного просмотра. Доступны 6 стилей - VisualStudio2005, Office2003, Office2007Blue, Office2007Silver, Office2007Black, VistaGlass. Стиль по умолчанию - Office2007Black.
`bool UseOffice2007Form`	Это свойство влияет на окна дизайнера и предварительного просмотра. Оно определяет, надо ли использовать окно в стиле Office2007, если выбран один из стилей интерфейса - Office2007Blue, Office2007Silver, Office2007Black, VistaGlass.Значение по умолчанию - true.

Кроме свойств, компонент EnvironmentSettings имеет несколько событий. Используя события, вы можете:

- 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 компонент EnvironmentSettings (см. раздел "Конфигурация среды FastReport.Net"). Он имеет следующие события:

СОБЫТИЕ	ОПИСАНИЕ
`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: `privatevoid 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;`` }``}`
`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: `privatevoid 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;`` }``}`
`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: `privatevoid CustomOpenReport_Handler(`` object sender, OpenSaveReportEventArgs e)``{`` // load the report from the given e.FileName`` e.Report.Load(e.FileName);``}`
`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: `privatevoid 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 ReportSettings.ShowProgress объекта EnvironmentSettings в false. Вы также можете подключить собственное окно прогресса. Для этого используйте следующие события, которые определены в классе EnvironmentSettings (см. раздел "Конфигурация среды FastReport.Net"):

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 компонента EnvironmentSettings (см. раздел "Конфигурация среды FastReport.Net"). 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:

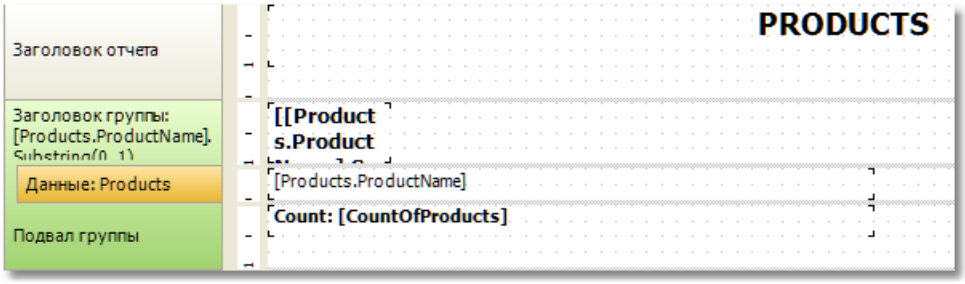
```
TextObject text1 = report1.FindObject("Text1") as TextObject;
text1.Font = new Font("Arial", 12);
```

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
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 компонент EnvironmentSettings (см. раздел "Конфигурация среды FastReport.Net"), вы можете настроить внешний вид и поведение стандартного окна просмотра. Соответствующие настройки доступны в свойстве EnvironmentSettings.PreviewSettings.

Если вас по каким-то причинам не устраивает стандартное окно просмотра, вы можете создать свое собственное. Для этого используйте элемент управления PreviewControl, 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 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 входит пример Demos\C#\CustomPreview, в котором показано, как управлять работой окна просмотра.

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 Config.DesignerSettings.FilterConnectionTables. Следующий пример показывает, как убрать из списка таблицу с именем "Table 1":

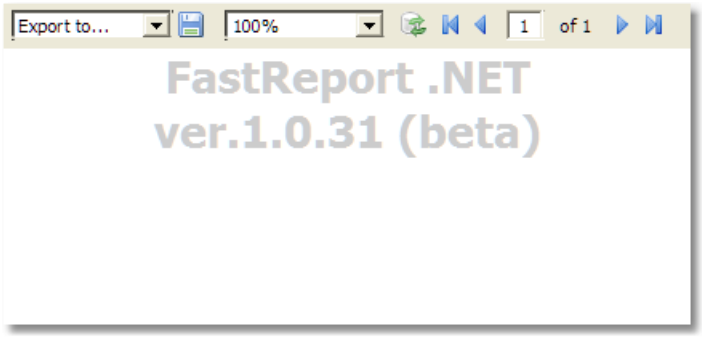
```
using FastReport.Utils;
Config.DesignerSettings.FilterConnectionTables += FilterConnectionTables;
private void FilterConnectionTables(
  object sender, FilterConnectionTablesEventArgs e)
{
  if (e.TableName == "Table 1")
    e.Skip = true;
}
```

Работа в ASP.NET

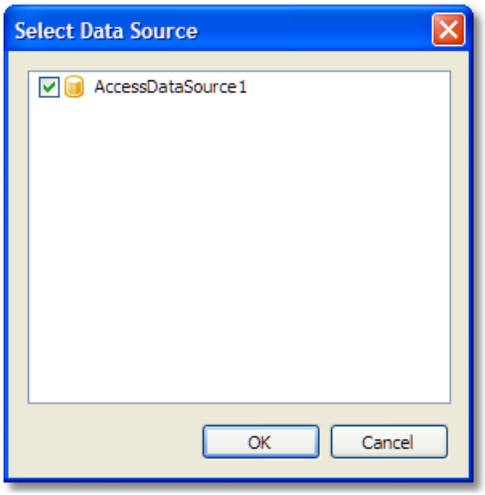
Использование компонента WebReport

Рассмотрим типичный сценарий использования компонента WebReport.

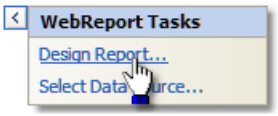
- предполагается, что вы имеете веб-проект, в котором настроены источники данных (например, AccessDataSource);
- положите компонент WebReport на вашу веб-форму:



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

При совместной работе с сервером IIS7 необходимы следующие строки:

```
<system.webServer><br/><handlers><br/><add name="FastReportHandler" path="FastReport.Export.axd" verb="*"
type="FastReport.Web.Handlers.WebExport"/><br/></handlers><br/></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><br/>      <add key="FastReportStoragePath" value="\\FS\WebReport_Exchange"/><br/>      <add
key="FastReportStorageTimeout" value="10"/><br/>      <add key="FastReportStorageCleanup" value="1"/><br/>
 </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"))<br/> {<br/>         <input id="pdf" type="submit" value="Export to PDF" /><br/> }
```

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()<br/>@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

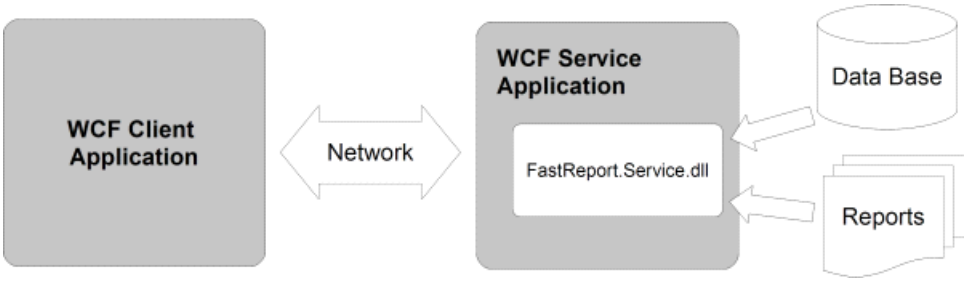
Простой пример WCF сервиса

Создание веб-сервиса на основе FastReport.Service.dll

Создание сервиса WCF на основе службы Windows

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
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.

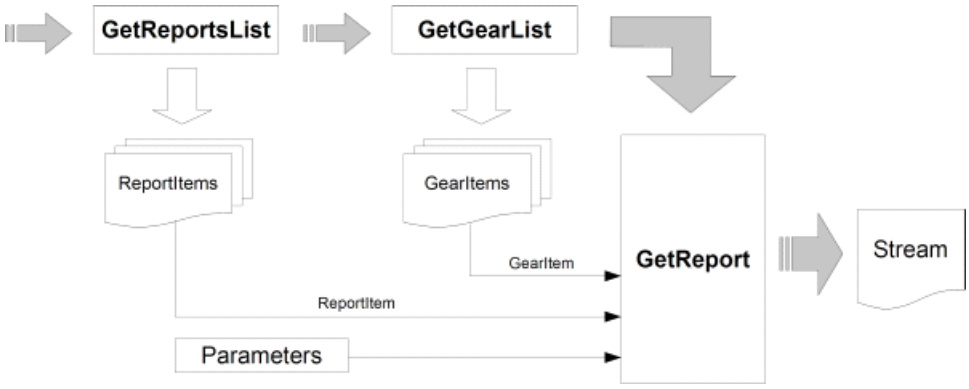
```
<appSettings>
<add key="FastReport.ReportsPath" value="C:\Program files\FastReports\FastReport.Net\Demos\WCF" />
<add key="FastReport.ConnectionStringName" value="FastReportDemo" />
<add key="FastReport.Gear" value="PDF,DOCX,XLSX,PPTX,RTF,ODS,ODT,MHT,CSV,DBF,XML,TXT,FPX" />
</appSettings>
```

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" />
```

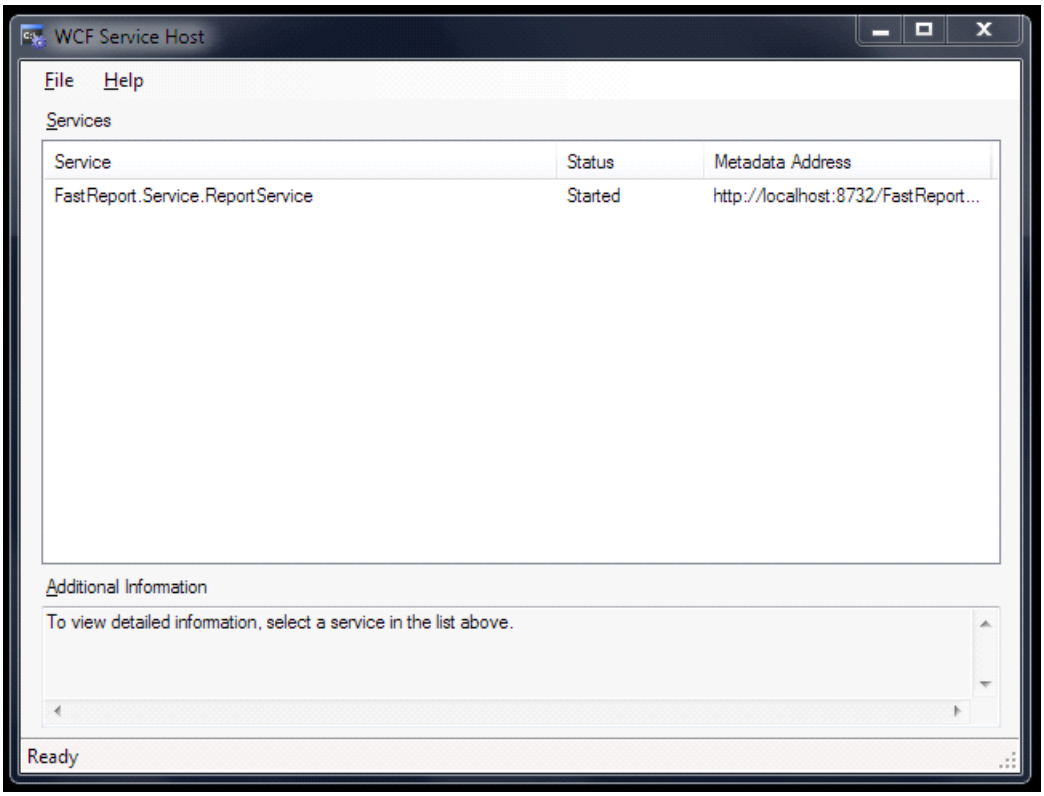
8. В секции меняем строку подключения к нашей базе:

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

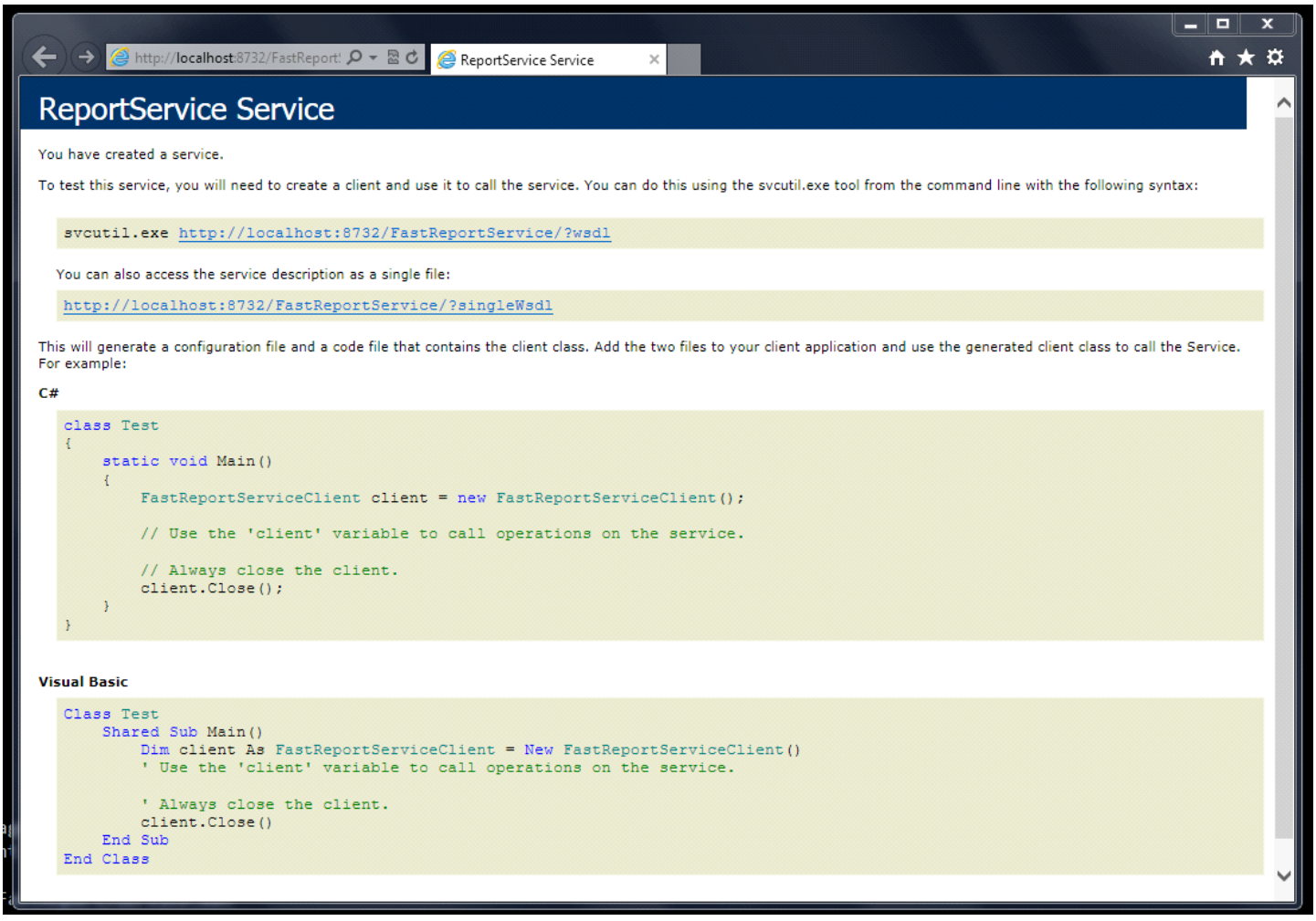
9. Создаем файл service.bat со следующей строкой:

"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/

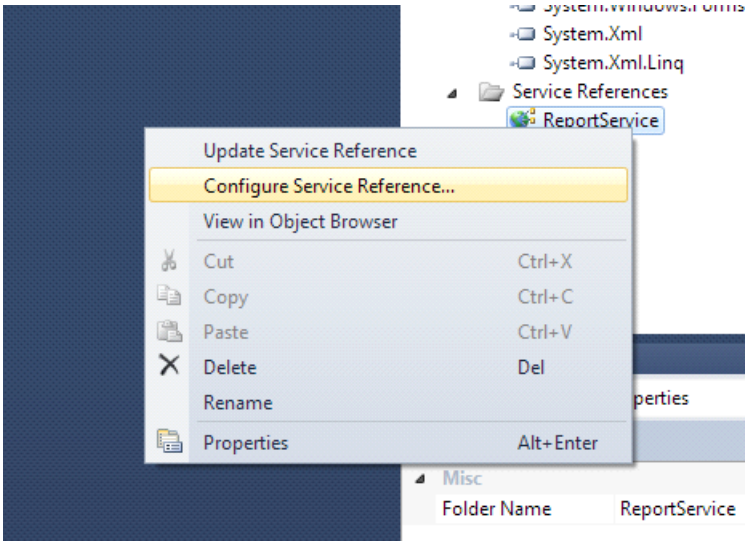


Номер порта сервиса можно настроить в файле конфигурации в строке

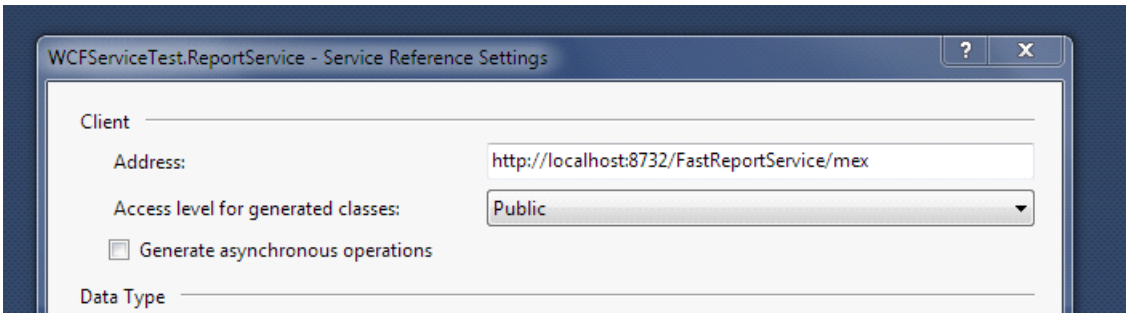
```
<add baseAddress="http://localhost:8732/FastReportService/" />
```

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

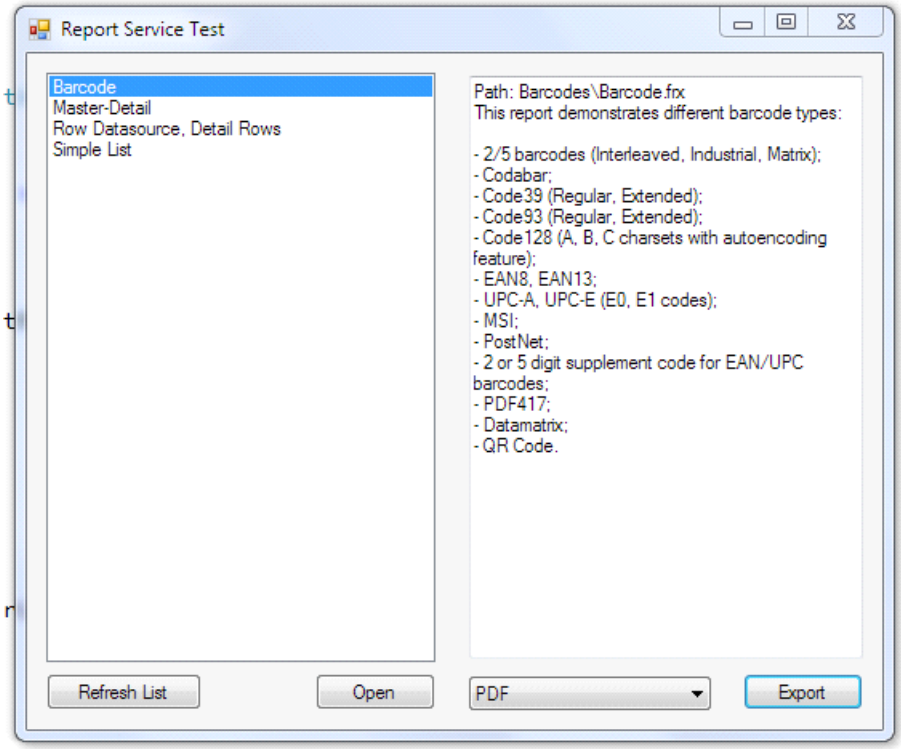
1. Открываем WCFServiceClient.csproj в Visual Studio
2. В дереве Solution Explorer кликаем правой кнопкой на Service References – ReportService и выбираем в меню Configure Service Reference



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



4. Компилируем и запускаем пример.

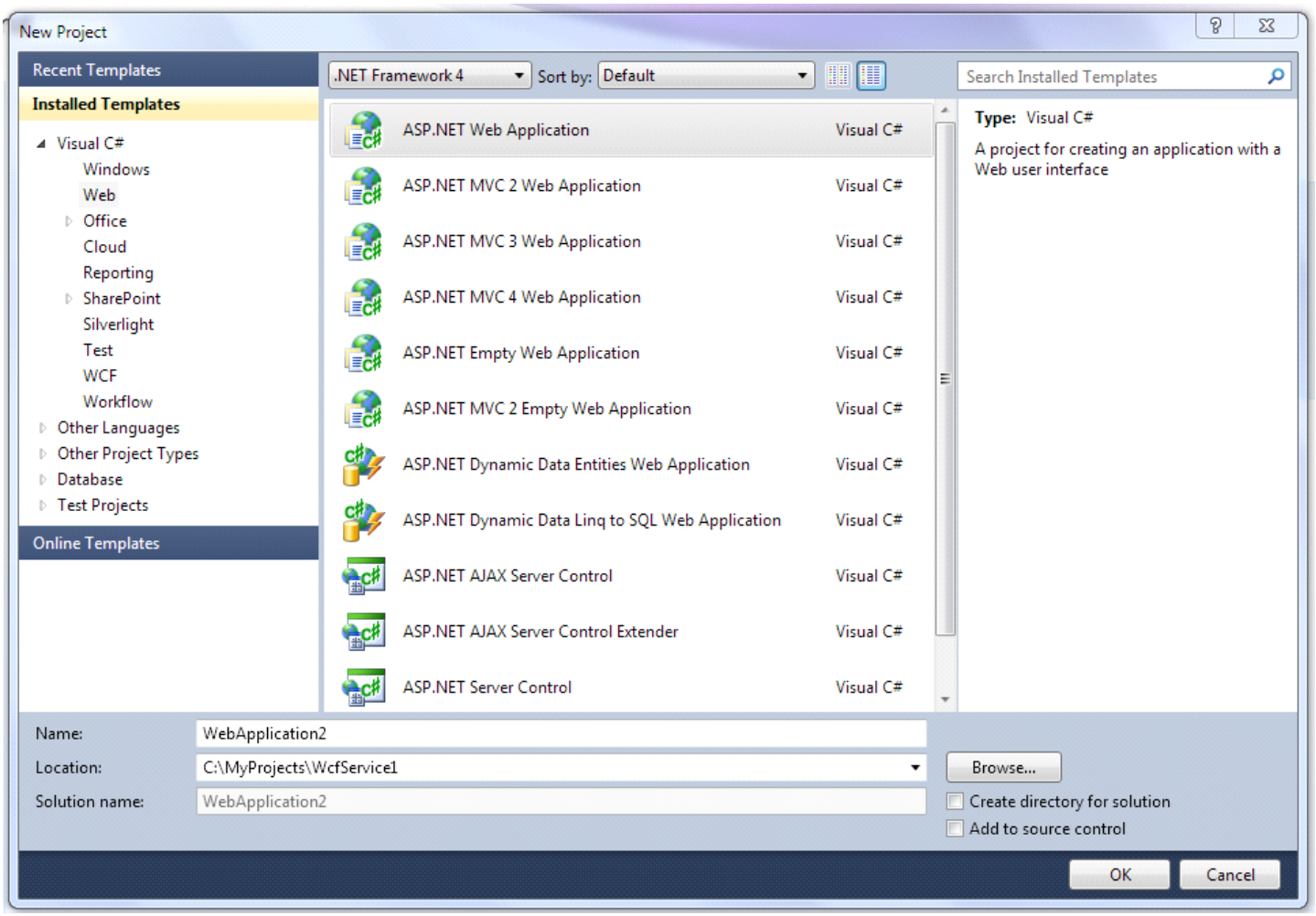

Report Service Test
Barcode
Master-Detail
Row Datasource, Detail Rows
Simple List
Path: Barcodes\Barcode.frx
This report demonstrates different barcode types:
- 2/5 barcodes (Interleaved, Industrial, Matrix);
- Codabar;
- Code39 (Regular, Extended);
- Code93 (Regular, Extended);
- Code128 (A, B, C charsets 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.

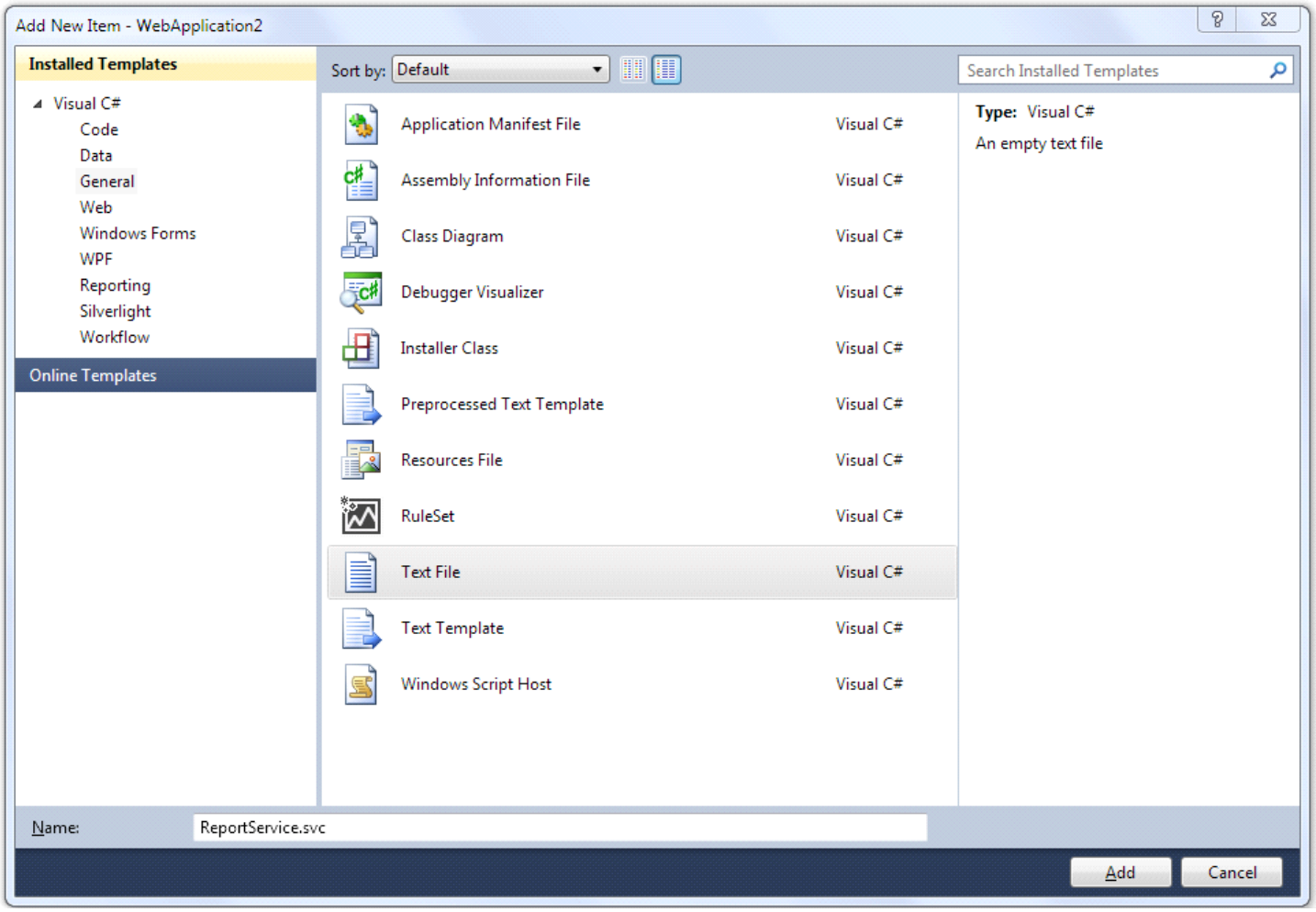
Пример основан на создании простейшего веб-приложения с функциями веб-сервиса с нуля, но так же можно доработать уже существующий ваш проект – главное, чтобы он работал под управлением .NET Framework 4.0 или более новым.

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" %>
```

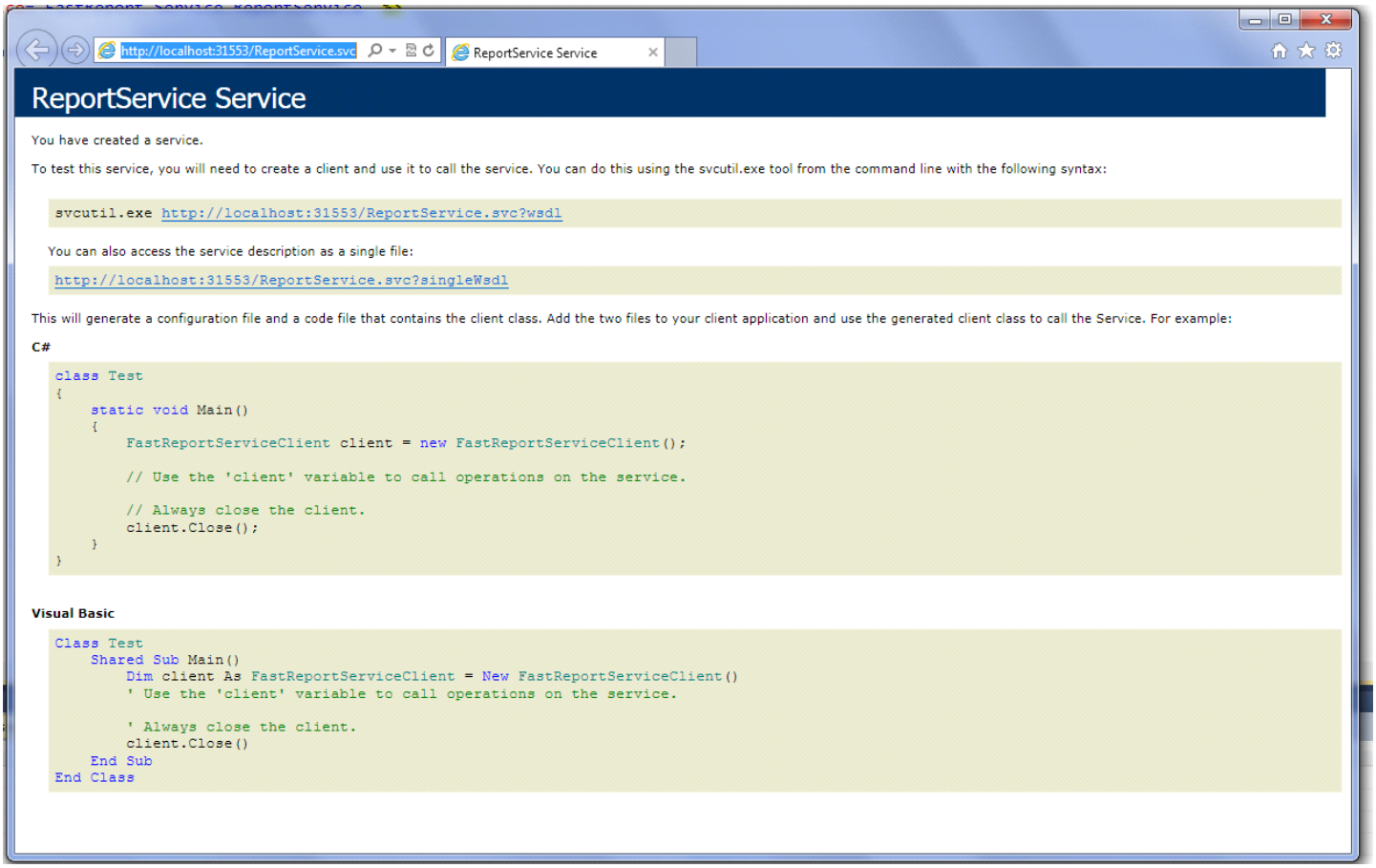
Открываем файл web.config и добавляем следующие секции в секцию :

```
<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».

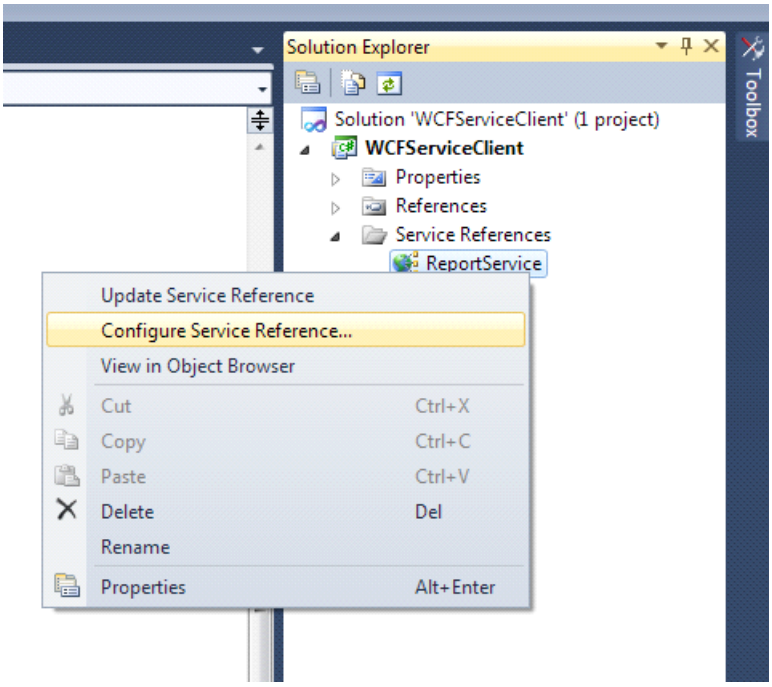
Параметр "FastReport.ConnectionStringName" содержит название строки подключения к базе данных. Эта строка должна быть прописана в секции .

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.

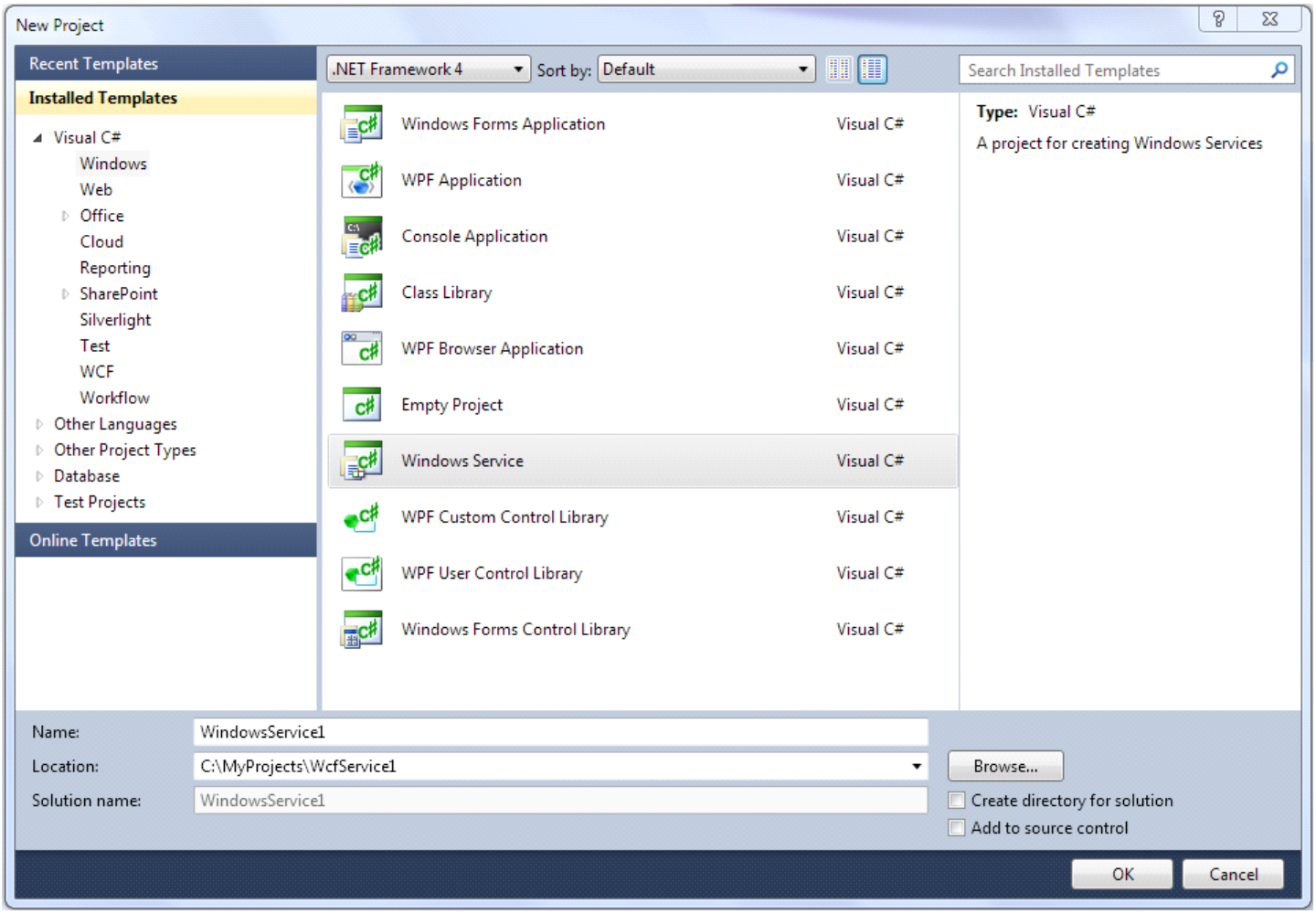
Примеры клиентских программ можно посмотреть в папках \FastReport.Net\Demos\C#\WCFClient и \FastReport.Net\Demos\C#\WCFWebClient. В каждом из примеров нужно сделать настройку Service References. Откройте проект в Visual Studio и по клику правой кнопкой мыши на ReportService выберете пункт меню Configure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



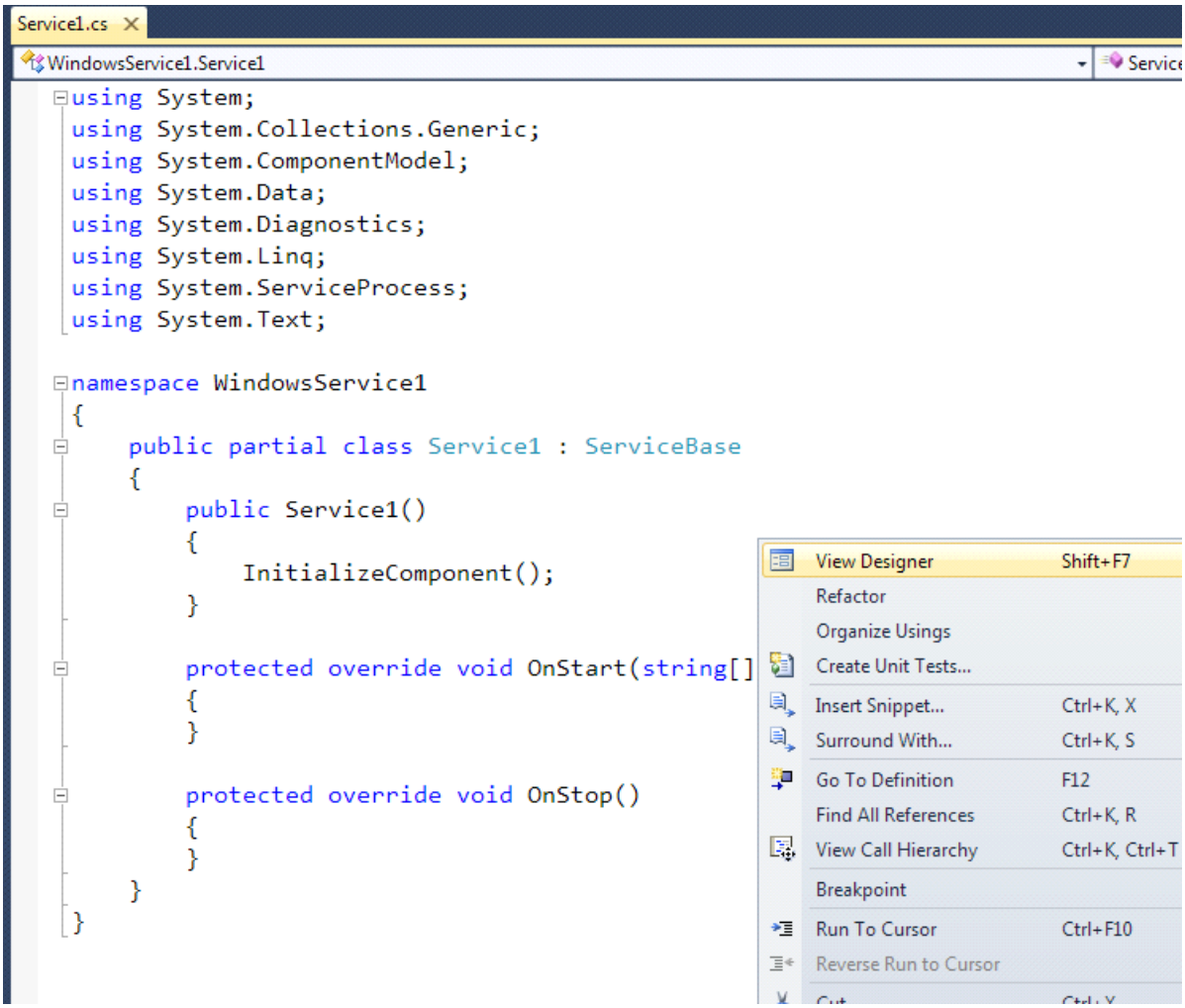
В открывшемся окне укажите адрес работающего веб-сервиса.

Создание сервиса WCF на основе службы Windows

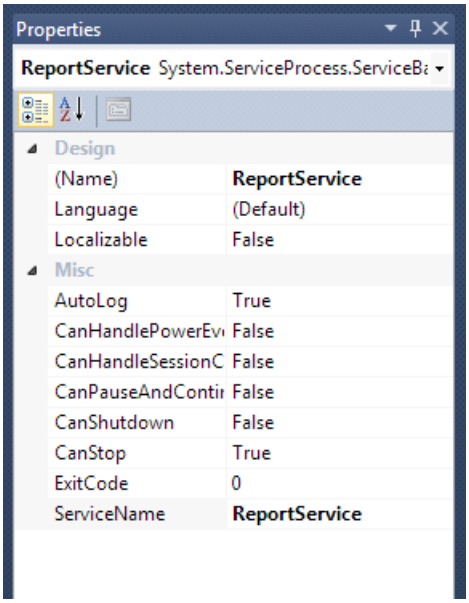
Нужно открыть Visual Studio и создать проект WindowsService.



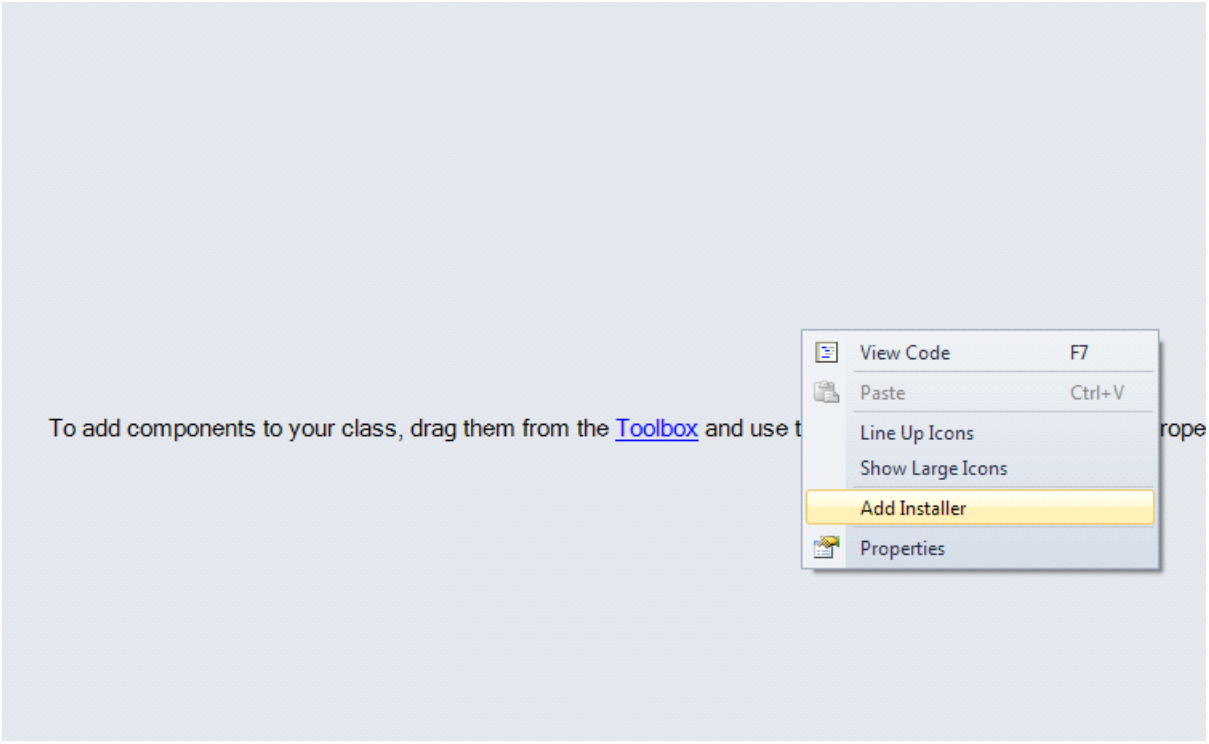
Открываем дизайнер Service1.cs



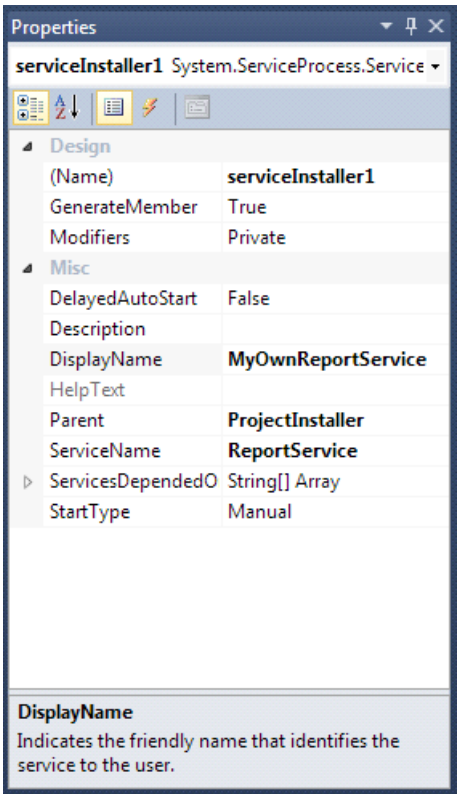
Изменяем имя сервиса по умолчанию на свое.



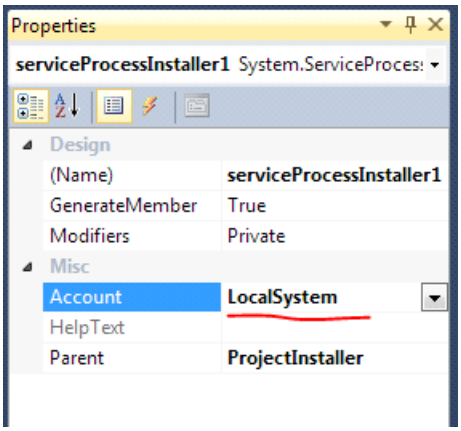
Кликаем по пустому окну дизайнера и выбираем пункт меню Add Installer.



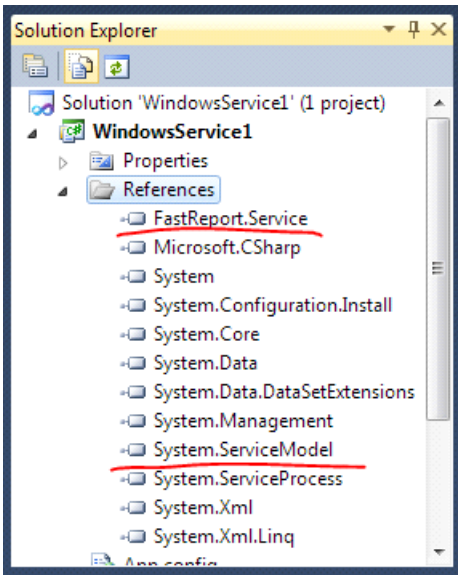
Редактируем свойства компонента serviceInstaller1 – указываем нужный DisplayName.



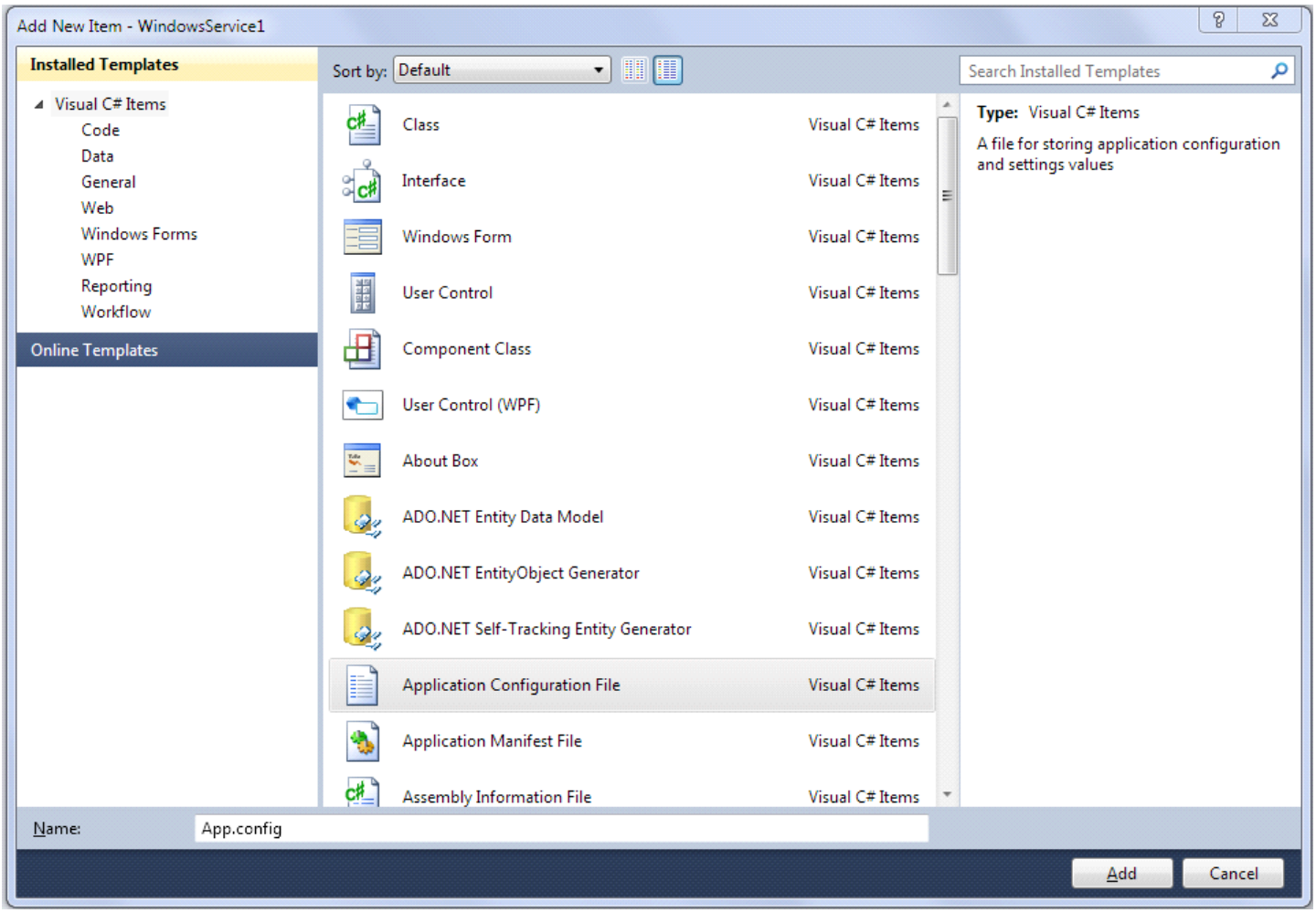
В свойствах компонента serviceProcessInstaller1 указываем тип аккаунта для работы службы LocalSystem.



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"

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.

В FastReport.Net на данный момент реализованы следующие типы подключений:

- MS Access (MsAccessDataConnection)
- OLE DB (OleDbDataConnection)
- MS SQL (MsSqlDataConnection)
- ODBC (OdbcDataConnection)
- подключение к локальным данным в формате .XML (XmlDataConnection).

Следующие типы подключений реализованы в виде подключаемых модулей (Add-in) и доступны для скачивания на странице http://www.fast-report.com/ru/download/fastreport.net-download.html:

- IBM DB2
- Firebird
- MySQL
- Oracle
- Postgres
- SQLite
- VistaDB

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

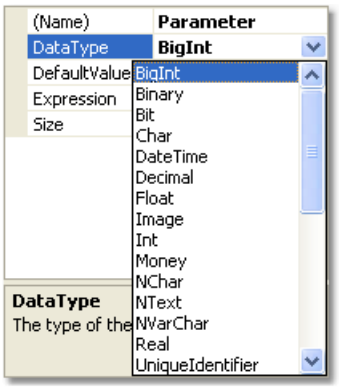
Этот метод возвращает экземпляр редактора для данного типа подключения. Реализация редактора будет рассмотрена в разделе "Редактор подключения".

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:



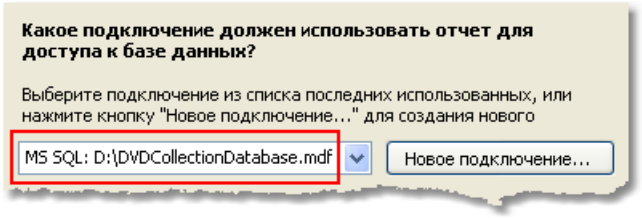
Например, для MsSqlDataConnection параметр имеет тип SqlDbType:

```
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, <br/>  CommandParameterCollection parameters)<br/>{<br/>  using (DbConnection conn = GetConnection())<br/>  {<br/>    OpenConnection(conn);<br/><br/>    // prepare select command<br/>    selectCommand = PrepareSelectCommand(selectCommand, table.TableName, conn);<br/><br/>    // read the table schema<br/>    using (DbDataAdapter adapter = GetAdapter(selectCommand, conn, parameters))<br/>    {<br/>      adapter.SelectCommand.CommandTimeout = CommandTimeout;<br/>      adapter.FillSchema(table, SchemaType.Source);<br/>    }<br/>  }<br/>}
```

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.

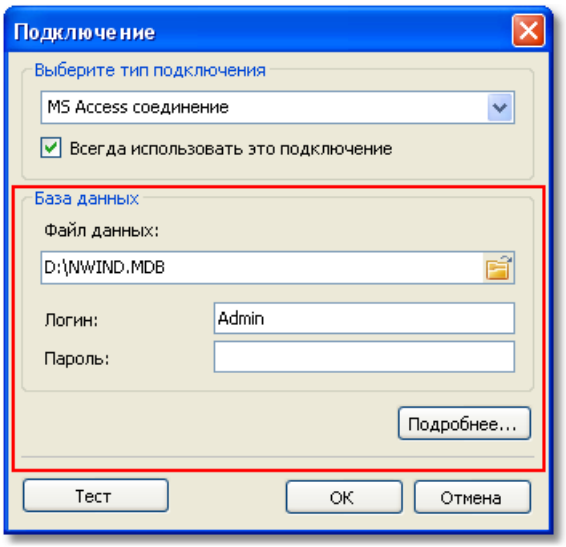
Метод имеет реализацию по умолчанию, которая использует объекты, возвращаемые методами GetConnection и GetAdapter:

```
public virtual void FillTableData(DataTable table, string selectCommand, <br/>  CommandParameterCollection parameters)<br/>{<br/>  using (DbConnection conn = GetConnection())<br/>  {<br/>    OpenConnection(conn);<br/><br/>    // prepare select command<br/>    selectCommand = PrepareSelectCommand(selectCommand, table.TableName, conn);<br/><br/>    // read the table<br/>    using (DbDataAdapter adapter = GetAdapter(selectCommand, conn, parameters))<br/>    {<br/>      adapter.SelectCommand.CommandTimeout = CommandTimeout;<br/>      table.Clear();<br/>      adapter.Fill(table);<br/>    }<br/>  }<br/>}
```

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):

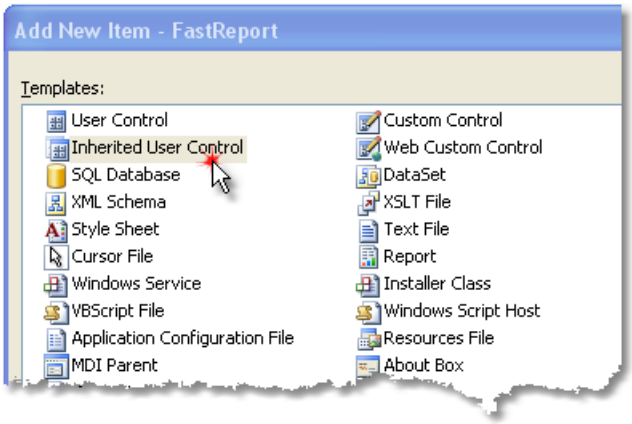


Для каждого типа подключения имеется свой редактор. Все редакторы наследуются от базового класса - FastReport.Data.ConnectionEditors.ConnectionEditorBase:

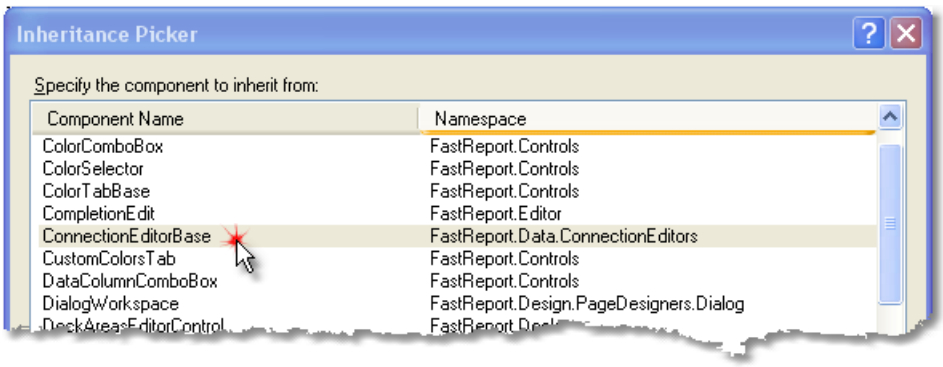
```
public class ConnectionEditorBase : UserControl
{
  protected virtual string GetConnectionString();
  protected virtual void SetConnectionString(string value);
}
```

Для создания своего редактора сделайте следующее:

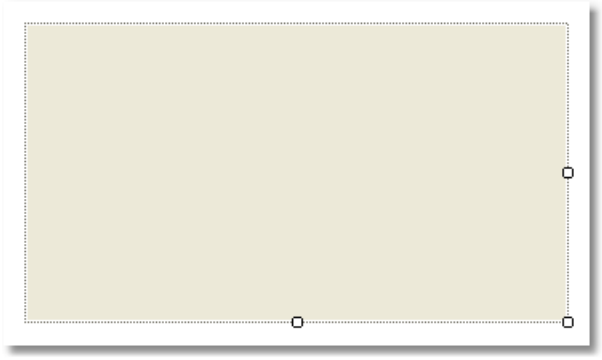
- добавьте в проект новый элемент управления (Add->User Control...) и выберите в окне "Add New Item" элемент "Inherited User Control":



- 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 "ОК". 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));
```

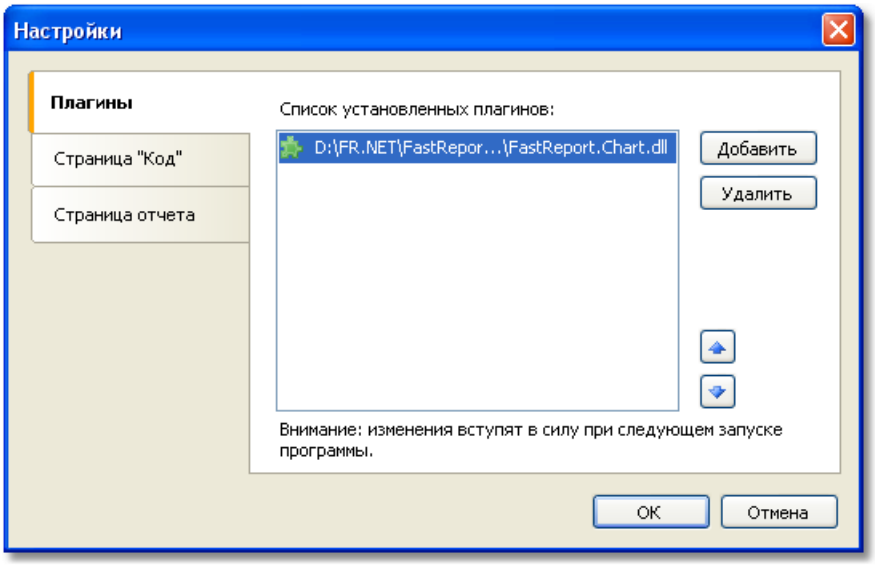
Этот код необходимо выполнить один раз за все время жизни приложения, перед запуском дизайнера.

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>
```