**Руководство по использованию библиотеки ExcelReportGenerator**

[Введение 3](#_Toc510903696)

[Шаблоны 3](#_Toc510903697)

[1. Шаблон свойства 3](#_Toc510903698)

[2. Шаблон вызова метода 5](#_Toc510903699)

[3. Шаблон элемента данных 6](#_Toc510903700)

[4. Шаблон агрегатной функции 7](#_Toc510903701)

[5. Шаблон системной переменной 8](#_Toc510903702)

[6. Шаблон вызова системной функции 8](#_Toc510903703)

[7. Шаблон горизонтального разрыва страницы 10](#_Toc510903704)

[8. Шаблон вертикального разрыва страницы 10](#_Toc510903705)

[Панели 10](#_Toc510903706)

[1. Простая панель 11](#_Toc510903707)

[2. Панель данных 13](#_Toc510903708)

[3. Панель итогов 20](#_Toc510903709)

[4. Динамическая панель 21](#_Toc510903710)

[5. Подключение и расширение 27](#_Toc510903711)

# Введение

Библиотека ExcelReportGenerator позволяет выводить данные в Microsoft Excel, размечая листы в нём определённым образом. В основе библиотеки лежат шаблоны и панели. Для взаимодействия с OpenXml используется библиотека [ClosedXml](https://github.com/ClosedXML/ClosedXML). Существует несколько типов шаблонов и панелей. Далее, рассмотрим как те, так и другие подробнее.

# Шаблоны

Шаблон представляет собой выражение, заключённое в некоторые границы. По умолчанию границами шаблона выступают фигурные скобки (возможна настройка), например, {p:Name} – вывод свойства или {m:GetData()} – вызов метода и т.п. Шаблон может занимать как ячейку целиком:



так и появляться в ячейке внутри текста:



Если шаблон находится внутри текста, то значение, которое он возвращает, всегда приводится к строке, то есть на нём вызывается метод ToString(). Если же шаблон занимает всю ячейку, то метод ToString() не вызывается, а значит в ячейку попадает значение того типа, которое вернул шаблон. Это особенно актуально для числовых типов, типов DateTime, Boolean и некоторых других.

Рассмотрим более подробно виды шаблонов.

## Шаблон свойства

Шаблон свойства в простейшем варианте имеет вид:

{p:PropName}

где p (property) – обозначает обращение к свойству, PropName – имя свойства (с учётом регистра). Свойство может быть как статическим, так и экземплярным, при этом оно обязательно должно иметь модификатор доступа public. Вместо имени свойства также может быть указано и имя публичного поля.

Данный шаблон может иметь более сложный вид. В частности можно склеивать обращения к свойствам (или полям) через «.», в таком случае шаблон примет следующий вид:

{p:Prop1.Prop2.….PropN}

Это будет означать, что нужно получить значение свойства (или поля) Prop1, далее вызвать на полученном значении свойство (или поле) Prop2 и т.д. При этом, если какое-то из свойств (или полей) в цепочке окажется равным null (кроме последнего), то будет выброшено исключение NullReferenceException. Свойств (или полей) в цепочке может быть сколько угодно.

По умолчанию свойство (или поле) ищется в типе отчёта, экземпляр которого передаётся конструктору класса DefaultReportGenerator или производного от него. Но это поведение можно переопределить, указав перед именем свойства тип, например:

{p:Company:Name}

В данном случае свойство (или поле) Name будет искаться в типе Company (тип отделяется от имени свойства символом «:»). При этом сам тип по умолчанию ищется в сборке исполняемого файла. Если нужный вам тип находится в другой сборке, то необходимо указать эту сборку в классе, производном от класса DefaultReportGenerator. В данном случае тип Company будет искаться во всех сборках, указанных в наследнике класса DefaultReportGenerator. Если будет найден только один тип с таким именем, то в нём будет произведён поиск свойства (или поля) Name. Если свойство (или поле) будет найдено, то шаблон вернёт его значение и поместит в соответствующее место на листе. Если же свойство в классе отсутствует, либо не соответствует модификаторам доступа (public, instance, static), то будет выброшено исключение MemberNotFoundException.

Если свойство не является статическим, то будет предпринята попытка создания экземпляра данного типа. По умолчанию в таком случае тип должен иметь конструктор без параметров. Это поведение можно переопределить в наследнике классе DefaultReportGenerator, предоставив свою реализацию интерфейса IInstanceProvider, которая, например, будет получать экземпляры объектов с помощью контейнера IoC. Стоит заметить, что в реализации по умолчанию, созданный экземпляр будет являться Singleton-объектом, то есть при множественных обращениях к экземплярным членам данного типа, они будут вызываться на объекте, созданном при первом обращении.

Если найдется более одного типа с данным именем, то будет выброшено исключение InvalidTemplateException. В данном случае можно скорректировать шаблон, указав в нём перед типом явно пространство имён, например:

{p:Reports.Common:Company:Name}

где Reports.Common – пространство имён, которое отделяется от имени самого типа с помощью «:». Отдельные пространства имён разделяются с помощью «.».

Также может быть ситуация, когда тип не будет найден. В этом случае будет выброшено исключение TypeNotFoundException.

Здесь стоит упомянуть ещё одну возможность, доступную для свойств и полей объекта, указанных в шаблоне. Если свойство или поле имеет значение null, и оно указано в шаблоне, то шаблон просто заменится на пустую строку. Если же вместо пустой строки вы хотите выводить какое-либо другое значение, то можно пометить данное свойство или поле атрибутом NullValue, передав ему в конструкторе необходимое значение, например:



В данном случае, если свойство Signer будет иметь значение null, то вместо пустой строки на месте шаблона выведется значение Smith. Значение, которое принимает атрибут, имеет тип object, то есть можно указывать любое значение, корректное для атрибута.

Данный шаблон можно использовать как во всех типах панелей (кроме динамической), так и просто на листе. Также данный шаблон можно передавать шаблону вызова метода в качестве параметра.

## Шаблон вызова метода

Шаблон вызова метода в простейшем варианте имеет вид:

{m:GetData()}

где m (method) – обозначает вызов метода, GetData() – непосредственно вызов самого метода. Метод может быть как статическим, так и экземплярным, при этом он обязательно должен иметь модификатор доступа public. Методу можно передавать сколько угодно параметров, причём в качестве параметров могут выступать как статические данные, зашитые внутри шаблона (строки, числа и т.п.), так и практически любые типы шаблонов. Ниже представлен вполне корректный вызов метода:

{m:GetData(p:Code, EN, m:Namespace:GetCompany(m:GetName(sv:Lang)), 56)}

Отсюда видно следующее:

* границы шаблона обрамляют только вызов самого внешнего метода, шаблоны внутри метода уже пишутся без границ;
* методу в качестве параметров можно передавать как статические данные, так и любые другие шаблоны, в том числе вызовы других методов (вложенность вызова методов не ограничена);
* как и в случае с шаблонами свойств, перед именем метода можно указать имя типа (включая пространство имён), поиск типа, и при необходимости создание экземпляра, в данном случае осуществляется точно так же, как и для шаблонов свойств;
* статические данные, даже если это строки, необязательно заключать в кавычки, при вызове метода будет предпринята попытка преобразовать переданные параметры к типам соответствующих параметров метода, если преобразование провалится, то будет выброшено соответствующее исключение.

Поиск метода осуществляется по имени (с учётом регистра), а также по количеству и типам параметров. Методы с переменным числом параметров (params) не поддерживаются. Могут возникнуть ситуации, особенно при перегрузке методов, когда необходимо будет указать в шаблоне тип статического параметра явно, это возможно сделать следующим способом:

{m:GetData([string]10, [decimal]56.6)}

то есть в данном случае перед параметром явно указывается его тип. Это актуально для всех примитивов, а также для типов String, Decimal и DateTime. Следует отметить, что для строк, вместо указания типа, можно просто обернуть параметр в кавычки, то есть написать "10", вместо [string]10.

Также возможна ситуация, когда внутри статического параметра присутствует запятая. По умолчанию запятые расцениваются как разделители параметров. Чтобы пометить запятую как простой символ, следует экранировать её ещё одной запятой, то есть написать «,,», например:

{m:GetData(p:Code, Hello,, World!)}

Данный шаблон можно использовать как во всех типах панелей (кроме динамической), так и просто на листе.

## Шаблон элемента данных

Шаблон элемента данных в простейшем варианте имеет вид:

{di:Amount}

где di (data item) – обозначает обращение к элементу данных, Amount – в зависимости от того, чем является элемент данных, может быть именем публичного поля или свойства (элемент данных – экземпляр класса), именем колонки (элемент данных – DataRow), ключом словаря (элемент данных IDictionary). В случае если элементом данных является экземпляр класса, то возможно склеивание свойств или полей через «.». Так как панели могут вкладываться друг в друга, то в данном шаблоне можно получить доступ к текущему элементу данных родительской панели следующим образом:

{di:parent:Code}

что будет означать, что нужно получить поле Code из элемента данных родительской панели. Также, если элемент данных имеет простейший тип, например int или string, и вы хотите вывести сам этот элемент, то в таком случае обращение должно быть следующим:

{di:di}

Если элементом данных является экземпляр класса, то к его свойствам и полям возможно применение атрибута NullValue.

Данный шаблон можно использовать только внутри панелей, которые имеют свойство DataSource (кроме динамической). В панели итогов – только в составе агрегатной функции, либо при обращении к элементу родительской панели. Также данный шаблон можно передавать шаблону вызова метода в качестве параметра.

## Шаблон агрегатной функции

Шаблон агрегатной функции в простейшем варианте имеет вид:

{Sum(di:Amount)}

где Sum – имя агрегатной функции, di:Amount – элемент данных, по которому следует произвести агрегацию. Если элемент данных представляет из себя экземпляр класса, то обращение к свойствам можно склеивать через «.».

Поддерживаются следующие встроенные агрегатные функции: Sum, Count, Avg, Min, Max. Также есть возможность создать свою агрегатную функцию, в таком случае в шаблоне нужно указать функцию Custom и вторым параметром наименование этой функции. При этом сама функция должна размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. То есть шаблон в данном случае будет иметь вид:

{Custom(di:Amount, CustomAggregationFunc)}

При этом CustomAggregationFunc должна иметь следующую сигнатуру:



где TResult – тип возвращаемого результата, TAccumulation – тип накапливаемого результата, TItem - тип элемента данных, itemNumber – номер текущего элемента данных (первый элемент будет иметь значение «1»). То есть на каждой итерации в функцию передаётся накопленный результат, текущий элемент данных и номер текущего элемента данных. Например, для вычисления произведения, это может выглядеть так:



Также есть возможность указать третьим параметром функцию, которая отработает после агрегации:

{Custom(di:Amount, CustomAggregationFunc, PostAggregationFunc)}

где PostAggregationFunc – имя функции, которая вызовется после агрегации. Данная функция должна размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура функции должна выглядеть следующим образом:



где TResult – тип возвращаемого результата, TAggregationResult – тип результата агрегации, aggregationResult – результат агрегации, itemsCount – количество элементов данных, например:



Если необходимо применить функцию постагрегации для предопределённой агрегатной функции, то можно просто опустить второй параметр, в таком случае шаблон примет вид:

{Min(di:Amount, , PostAggregationFunc)}

Данный шаблон можно использовать только внутри панелей итогов. Также данный шаблон можно передавать шаблону вызова метода в качестве параметра.

## Шаблон системной переменной

Шаблон вывода системной переменной имеет вид:

{sv:SheetName}

где sv (system variable) – обозначает обращение к системной переменной, SheetName – имя переменной (с учётом регистра). В настоящий момент доступны следующие предопределённые системные переменные:

* RenderDate – дата рендеринга (тип DateTime);
* SheetName – имя листа (тип String);
* SheetNumber – номер листа (тип int).

Существует возможность расширения системных переменных. Данный функционал будет описан в отдельной главе.

Данный шаблон можно использовать как во всех типах панелей (кроме динамической), так и просто на листе. Также данный шаблон можно передавать шаблону вызова метода в качестве параметра.

## Шаблон вызова системной функции

Шаблон вызова системной функции имеет вид:

{sf:Format(p:value, "C")}

где sf (system function) – обозначает вызов системной функции, Format(p:value, "C") – непосредственно вызов самой функции. В настоящий момент доступны следующие предопределённые системные функции:



Данная функция позволяет получить значение из словаря по ключу. Параметр dictionary должен реализовывать интерфейс IDictionary. Если какой-либо из параметров окажется равным null, или параметр dictionary не реализует интерфейс IDictionary или данный ключ отсутствует в словаре, то будет выброшено соответствующее исключение.



Данная функция полностью аналогична функции GetDictVal за исключением того, что она никогда не выбрасывает исключение. В случае любой ошибки функция возвратит null.



Данная функция позволяет получить элемент списка по индексу. Параметр list должен реализовывать интерфейс IList. Если параметр list равен null или он не реализует IList или индекс выходит за границы диапазона, то будет выброшено соответствующее исключение.



Данная функция полностью аналогична функции GetByIndex за исключением того, что она никогда не выбрасывает исключение. В случае любой ошибки функция возвратит null.



Данная функция позволяет преобразовать входное значение в строку с заданным форматированием. Параметр input должен реализовывать интерфейс IFormattable, в противном случае будет выброшено исключение. Также существует третий необязательный параметр для возможности задания культуры. В этот параметр может быть передан объект одного из трёх типов:

* IFormatProvider – применяется как есть.
* string – в данном случае это должно быть наименование культуры ("en", "en-US", "ru", "fr" и т.д.). Будет предпринята попытка создания объекта CultureInfo с этим наименованием. Если культура не будет найдена, то будет выброшено исключение CultureNotFoundException.
* int – в данном случае это должен быть код культуры, например, 9 для культуры "en" или 1049 для культуры "ru-RU". Аналогично будет предпринята попытка создания объекта CultureInfo с этим кодом. Если культура не будет найдена, то будет выброшено исключение CultureNotFoundException. Стоит заметить, что при передаче кода культуры напрямую из Excel, нужно явно указать в шаблоне тип int, например:

{sf:Format(p:value, "C", [int]1033)}

Системные функции можно расширять. Данный функционал будет описан в отдельной главе.

Данный шаблон можно использовать как во всех типах панелей (кроме динамической), так и просто на листе. Также данный шаблон можно передавать шаблону вызова метода в качестве параметра.

## Шаблон горизонтального разрыва страницы

Шаблон горизонтального разрыва страницы имеет вид:

{HorizPageBreak}

Данные шаблон позволяет вставить разрыв страницы после строки, в которой находится ячейка с данным шаблоном. Это может быть полезно при вложении панелей данных друг в друга, если вы хотите каждый блок данных выводить на печать на отдельной странице.

## Шаблон вертикального разрыва страницы

Шаблон вертикального разрыва страницы имеет вид:

{VertPageBreak}

Данные шаблон позволяет вставить разрыв страницы после колонки, в которой находится ячейка с данным шаблоном.

# Панели

Панели представляют собой именованные диапазоны ячеек в Microsoft Excel, которые называются определённым образом. Например, на рисунке ниже представлена простая панель с именем s\_Sample и координатами (B2, C3).



Чтобы именованный диапазон превратился в панель, имя должно формироваться по принципу <prefix>\_<name>, где <prefix> обозначает тип панели, <name> – имя панели, которое может быть любым. Какие префиксы имеют различные типы панелей, будет описано ниже.

Именованные диапазоны в Microsoft Excel могут иметь более одного диапазона ячеек. Например, на рисунке ниже, именованный диапазон s\_Sample имеет координаты (B2, C3; D5, E5).



Библиотекой ExcelReportGenerator множественные диапазоны не поддерживаются, то есть в данном случае будет обработан только первый диапазон ячеек, а второй будет проигнорирован.

При создании именованного диапазона, кроме наименования и координат, можно задать также область и комментарий, как показано на рисунке ниже:



Область на рендеринг не оказывает никакого значения, а вот поле комментария необходимо для задания свойств панелей. По умолчанию, свойства могут разделяться либо символом новой строки, либо точкой с запятой, либо символом табуляции. При этом имя и значение свойства разделяются символом «=». Какие свойства может иметь каждая из панелей будет рассмотрено подробнее в соответствующих главах.

Панели могут вкладываться друг в друга, то есть иметь отношение родитель-потомок. В данном случае диапазон ячеек дочерней панели обязательно должен находиться внутри диапазона ячеек родительской панели, а также у дочерней панели в поле комментария должно быть указано свойство ParentPanel, равное имени родительской панели.

Сам лист также является панелью, поэтому неспецифичные для каких-либо конкретных панелей шаблоны можно писать прям на листе. Панель листа является родительской для всех остальных и всегда обрабатывается с наивысшим приоритетом.

## Простая панель

Наименование простой панели имеет вид:

s\_<name>

где s (simple) – обозначает, что это простая панель, <name> – имя панели, которое может быть любым, например s\_ReportHeader – корректное имя для простой панели. Чаще всего простые панели не должны явно обозначаться на листе, так как данные панели не обрабатываются каким-то особым образом, все шаблоны просто выводятся как есть, и следовательно их можно написать на самом листе, который тоже, по сути, является простой панелью. Смысл в данной панели может появиться, если вы хотите задать для неё какие-нибудь дополнительные свойства. Для данной панели актуальны следующие свойства:

1. RenderPriority (тип int) – устанавливает приоритет обработки панели. По умолчанию все панели имеют один и тот же приоритет, равный «0», но при необходимости его можно повысить, например, когда одна панель зависит от другой. При этом родительские панели всегда обрабатываются раньше потомков, какой бы приоритет не стоял у потомка, поэтому панель листа всегда обрабатывается самой первой, так как является родительской для всех остальных, и понизить её приоритет нельзя. То есть данное свойство не является сквозным, а влияет на обработку панелей одного уровня вложенности.
2. BeforeRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом панели. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где BeforeRender – имя метода, которое указывается в поле комментария для свойства BeforeRenderMethodName, args – параметр метода, имеющий тип PanelBeforeRenderEventArgs, который содержит в себе следующие поля:

* Range (тип ClosedXML.Excel.IXLRange) – диапазон ячеек, которые охватывает панель до рендеринга;
* IsCanceled (тип bool) – флаг, который позволяет отменить рендеринг панели; если внутри метода выставить его в true, то данная панель, а также все её потомки не будут обрабатываться библиотекой.

1. AfterRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга панели. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где AfterRender – имя метода, которое указывается в поле комментария для свойства AfterRenderMethodName, args – параметр метода, имеющий тип PanelEventArgs, который содержит в себе следующие поля:

* Range (тип ClosedXML.Excel.IXLRange) – диапазон ячеек, которые охватывает панель после рендеринга.

1. ParentPanel (тип string) – позволяет задать имя родительской панели. В этом случае диапазон ячеек родительской панели обязательно должен охватывать диапазон ячеек дочерней панели, иначе на этапе рендеринга возникнет исключение. Вряд ли понадобится делать простую панель вложенной. Смысл этого поля в полной мере раскроется при рассмотрении панелей с данными.

Данная панель может содержать в себе все типы шаблонов. При этом шаблон элемента данных будет корректно работать только, если данная панель имеет в качестве родителя панель данных.

## Панель данных

Наименование панели данных имеет вид:

d\_<name>

где d (data) – обозначает, что это панель данных, <name> – имя панели, которое может быть любым, например d\_ReportData – корректное имя для панели данных. Данная панель может содержать практически любые типы шаблонов, но специфичным для неё является шаблон элемента данных, который имеет вид {di:Field}, где Field может обозначать разное в зависимости от типа источника данных. Сам шаблон элемента данных более конкретно описан в соответствующей главе. Данная панель имеет обязательное свойство DataSource, в котором должен указываться какой-либо шаблон, возвращающий источник данных. Данная панель отобразится на листе столько раз, сколько элементов содержит в себе источник данных. Если источник данных не содержит элементов, то панель будет удалена (удаление происходит со сдвигом, указанным в поле ShiftType). В качестве источника данных могут выступать следующие типы:

1. DataTable – элементом данных выступает объект DataRow, в шаблоне указывается имя колонки.
2. DataSet – элементом данных выступает объект DataRow, так как из объекта DataSet извлекается первая таблица данных DataTable, соответственно в шаблоне указывается имя колонки первой таблицы объекта DataSet.
3. IDataReader – элементом данных также выступает объект DataRow, так как за кулисами IDataReader считывается в объект DataTable и сразу закрывается. Чтобы при закрытии объекта IDataReader, также закрылся и Connection к базе данных, при его считывании необходимо указать это явным образом:



В шаблоне указывается имя необходимого столбца, возвращаемого SQL-запросом.

1. IDictionary<TKey, TValue> – элементом данных выступает объект KeyValuePair<TKey, TValue>, в шаблоне можно обратиться либо к свойству Key, либо к свойству Value данной структуры (возможно склеивание свойств через «.»).
2. IEnumerable<IDictionary<string, TValue>> – элементом данных выступает словарь IDictionary<string, TValue>; все словари в последовательности должны быть одинаковы по структуре; в шаблоне указывается ключ словаря.
3. IEnumerable или IEnumerable<T> – элементом данных выступает объект, содержащийся в данной последовательности; в шаблоне указываются публичные поля или публичные свойства данного объекта (возможно склеивание полей или свойств через «.»).
4. Объект любого другого типа – элементом данных выступает сам этот объект, в шаблоне указываются публичные поля или публичные свойства данного объекта (возможно склеивание полей или свойств через «.»).

Для данной панели актуальные следующие свойства:

1. DataSource (обязательное, тип – любой, уместный в данном контексте шаблон). Шаблон должен возвращать один из источников данных, описанный выше. Шаблон в данном случае задаётся без границ, например, на рисунке ниже в качестве источника данных указан шаблон вызова метода:



Если данная панель является дочерней относительно другой панели данных, то она имеет доступ к контексту родительской панели, то есть может использовать в свойстве DataSource шаблоны элементов данных родительской панели. Например, на рисунке ниже шаблон элемента данных родительской панели указан в качестве источника данных для дочерней панели:



Шаблон di:Products здесь может являться, например, свойством объекта, содержащего некую коллекцию продуктов.

Также можно использовать шаблон элемента данных родительской панели в качестве параметра шаблона вызова метода, например:



Здесь шаблон элемента данных родительской панели di:Category передаётся в качестве параметра шаблону вызова метода. Всё это позволяет делать довольно сложные группировки данных. Функционал дочерних панелей данных более подробно будет описан далее.

1. RenderPriority (тип int) – полностью аналогично одноимённому свойству для простой панели.
2. Type (тип enum) – позволяет задать тип панели, может принимать одно из двух значений: Vertical (значение по умолчанию) и Horizontal. Это свойство влияет на то, в какую сторону будет расширяться панель данных (вниз или вправо). По умолчанию это расширение происходит вниз, то есть каждый новый элемент данных появляется под предыдущим. Это поведение можно изменить, задав для данного свойства значение Horizontal. В таком случае расширение будет происходить вправо, то есть каждый последующий элемент данных будет появляться справа от предыдущего.
3. ShifType (тип enum) – позволяет задать тип сдвига при расширении панели данных. Может принимать одно из трёх значений: Cells (значение по умолчанию), Row и NoShift. Если установлено значение Cells, то для каждого последующего элемента данных будут выделяться новые ячейки в рамках диапазона, который занимает данная панель. То есть, для вертикальной панели, все ячейки, находящиеся под этим диапазоном будут смещены вниз, остальные останутся на своём месте. Для горизонтальной панели аналогично, только ячейки будут смещаться вправо. Пусть, например, существует следующая вертикальная панель данных с координатами (B2, C2), под которой и справа от которой размещаются некоторые текстовые данные:



В случае сдвига Cells, после рендеринга, текст «Bottom Text» сдвинется вниз на количество элементов данных минус один, а текс «Right text» останется на своём месте. Если предположить, что источник данных имеет три строки, то в результате получится следующее:



Если установлено значение Row, то для вертикальной панели для каждого последующего элемента данных будут выделяться новые строки целиком, независимо от того, сколько ячеек занимает сама панель. Для горизонтальной панели аналогично, только вместо строк будут выделяться колонки. В таком случае, для вертикальной панели, всё, что находится ниже неё, будет смещено вниз на количество элементов данных минус один. Касаемо предыдущего примера, результат будет следующий:



Если установлено значение NoShift, то ни дополнительные ячейки, ни дополнительные строки не выделяются, соответственно никаких сдвигов не происходит. Панель данных просто затирает ячейки под собой (для вертикальной панели) или справа от себя (для горизонтальной). Касаемо предыдущего примера, результат будет следующий:



Стоит заметить, что сдвиг NoShift нельзя задавать для дочерних панелей данных, так как в таком случае произойдёт наложение родительской и дочерней панели при рендеринге.

1. GroupBy (тип string) – позволяет задать номера колонок или строк (в случае горизонтальной панели), по которым будет осуществлена группировка, то есть ячейки с одинаковым значением, идущие подряд, в данной колонке или строке будут объединены. Номер – это порядковый номер колонки или строки в рамках диапазона панели, начинающийся с единицы. Можно перечислить несколько номеров колонок или строк, разделив их запятой, например: 2,4.
2. BeforeRenderMethodName (тип string) – аналогично одноимённому свойству для простой панели, за исключением того, что сигнатура метода должна быть следующей:



где тип DataSourcePanelBeforeRenderEventArgs помимо свойств Range и IsCanceled, также имеет ещё свойство Data (тип object). То есть перед рендерингом панели можно произвести какие-то дополнительные манипуляции с источником данных для данной панели.

1. AfterRenderMethodName (тип string) – аналогично одноимённому свойству для простой панели, за исключением того, что сигнатура метода должна быть следующей:



где тип DataSourcePanelEventArgs помимо свойства Range, также имеет ещё свойство Data (тип object).

1. BeforeDataItemRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом каждого элемента данных панели. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где BeforeRenderDataItem – имя метода, которое указывается в поле комментария для свойства BeforeDataItemRenderMethodName, args – параметр метода, имеющий тип DataItemPanelBeforeRenderEventArgs, который содержит в себе следующие поля:

* Range (тип ClosedXML.Excel.IXLRange) – диапазон ячеек, которые охватывает панель элемента данных до рендеринга;
* IsCanceled (тип bool) – флаг, который позволяет отменить рендеринг элемента данных;
* DataItem (тип HierarchicalDataItem) – непосредственно сам элемент данных (с учётом иерархичности, если панель данных вложена в другую панель данных).

1. AfterDataItemRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга каждого элемента данных панели. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где AfterRenderDataItem – имя метода, которое указывается в поле комментария для свойства AfterDataItemRenderMethodName, args – параметр метода, имеющий тип DataItemPanelEventArgs, который содержит в себе следующие поля:

* Range (тип ClosedXML.Excel.IXLRange) – диапазон ячеек, которые охватывает панель элемента данных после рендеринга;
* DataItem (тип HierarchicalDataItem) – непосредственно сам элемент данных (с учётом иерархичности, если панель данных вложена в другую панель данных).

1. ParentPanel (тип string) – позволяет задать имя родительской панели. В этом случае диапазон ячеек родительской панели обязательно должен охватывать диапазон ячеек дочерней панели, иначе на этапе рендеринга возникнет исключение. Также здесь стоит учесть, что для корректного сдвига родительской панели относительно дочерней, дочерняя панель должна занимать всю ширину родителя, если она является вертикальной или всю высоту родителя, если она является горизонтальной. При этом ненужные для рендеринга ячейки можно просто оставить пустыми. Также корректного сдвига можно добиться, даже если дочерняя панель не будет занимать всю ширину или высоту родителя, если для неё задать ShiftType=Row.

Рассмотрим более подробно вложение одной панели данных в другую. Представим, что мы хотим вывести продукты, сгруппированные по категориям. Для этого мы можем создать на листе две панели данных. Для родительской панели в качестве источника данных мы укажем коллекцию всех категорий, а для дочерней – укажем метод, которому будем передавать текущую категорию из родительской панели. На рисунке ниже показана разметка листа:



При этом в классе отчёта у нас должно быть свойство, возвращающее набор категорий, а также метод, который по коду категории возвращает набор продуктов. Вот так, к примеру, может выглядеть результат:



То есть вложенные панели позволяют выводить данные с группировкой.

В данном случае для дочерней панели мы использовали метод, который возвращал продукты по коду категории. Если бы каждая категория имела свойство, содержащее набор продуктов, то в качестве источника данных для дочерней панели можно было бы задать данное свойство, то есть разметка листа выглядела бы так:



Результат в итоге не изменится. Какой вариант использовать зависит от конкретной ситуации.

На данном примере можно также продемонстрировать горизонтальный разрыв страницы. Если в последнюю строку родительской панели вставить шаблон {HorizPageBreak}, то каждая категория будет выведена на печать на отдельной странице. В таком случае разметка панели примет вид:



Как уже упоминалось при рассмотрении шаблона элемента данных, из дочерней панели можно получить доступ к элементу данных родительской панели, используя в шаблоне указатель parent. Для предыдущего примера, чтобы получить доступ к коду категории из дочерней панели, нужно было бы написать:

{di:parent:CategoryCode}

Так как вложенность панелей не ограничена, то можно получить доступ к любому родителю в иерархии, тогда указатель parent нужно будет повторить необходимое количество раз.

Предыдущий пример, по сути, иллюстрирует группировку данных. Похожего по смыслу результата можно добиться также при помощи свойства панели данных GroupBy, хотя вывод будет немного отличаться. В данном случае дочерние панели будут не нужны, единственный источник данных должен возвращать сразу весь набор, отсортированный по категориям. Будет только одна панель данных со следующей разметкой:



Здесь указано, что группировка должна происходить по первой колонке, то есть в нашем случае по полю CategoryName. Результат при этом будет выглядеть примерно так:



Какой вариант группировки выбрать – через вложение панелей или посредством свойства GroupBy, зависит от того, какой результат вывода в итоге нужно получить.

## Панель итогов

Наименование панели итогов имеет вид:

t\_<name>

где t (totals) – обозначает, что это панель итогов, <name> – имя панели, которое может быть любым, например t\_ReportTotals – корректное имя для панели итогов. Данная панель может содержать практически любые типы шаблонов, но специфичным для неё является шаблон агрегатной функции, который в простейшем случае имеет вид {Sum(di:Field)}, где Sum – тип агрегатной функции, Field может обозначать разное в зависимости от типа источника данных. Сам шаблон агрегатной функции более подробно описан в соответствующей главе. Данная панель имеет обязательное свойство DataSource, в котором должен указываться какой-либо шаблон, возвращающий источник данных. Источники данных для данной панели полностью аналогичны источникам данных для панели данных.

Данная панель извлекает все агрегатные функции, указанные в ней, проходится в цикле по источнику данных и вычисляет значения данных агрегатных функций. Все агрегатные функции вычисляются за один проход по данным. Остальные шаблоны вычисляются обычным для себя образом. Стоит заметить, что шаблон элемента данных без агрегатной функции в данной панели может появляться только при обращении к элементу данных родительской панели (через указатель parent).

Для данной панели актуальные следующие свойства:

1. DataSource (обязательное, тип – любой, уместный в данном контексте шаблон) – полностью аналогично одноимённому свойству для панели данных.
2. RenderPriority (тип int) – полностью аналогично одноимённому свойству для простой панели.
3. BeforeRenderMethodName (тип string) – полностью аналогично одноимённому свойству для панели данных.
4. AfterRenderMethodName (тип string) – полностью аналогично одноимённому свойству для панели данных.
5. ParentPanel (тип string) – полностью аналогично одноимённому свойству для панели данных.

## Динамическая панель

Наименование динамической панели имеет вид:

dyn\_<name>

где dyn (dynamic) – обозначает, что это динамическая панель, <name> – имя панели, которое может быть любым, например dyn\_ReportData – корректное имя для динамической панели.

Динамическая панель отличается от всех остальных панелей. В ней используются свои, специфичные только для неё шаблоны. В полном варианте панель имеет следующий вид:



На рисунке выше представлены все виды шаблонов, которые может содержать динамическая панель. Любой из этих шаблонов можно опустить, а также они могут появляться в любом порядке. Данная панель имеет обязательное свойство DataSource. Далее рассмотрим каждый из шаблонов панели подробнее.

1. Шаблон {Headers} превращается в заголовки колонок (если панель вертикальная) или в заголовки строк (если панель горизонтальная). То есть в случае вертикальной панели, шаблон {Headers} размножается вправо, превращаясь в заголовки колонок, в случае горизонтальной, он размножается вниз. В зависимости от источника данных названия колонок или строк получаются по-разному:

* DataTable – названия колонок берутся из свойства Caption объекта DataColumn, если свойство Caption равно null, то названия колонок берутся из свойства Name. Тип колонок определяется из свойства DataType объекта DataColumn.
* DataSet – аналогично источнику DataTable, так как из объекта DataSet извлекается первая таблица данных DataTable.
* IDataReader – названия колонок – это имена столбцов, возвращаемых SQL-запросом. Тип колонок определяется из таблицы метаданных для объекта IDataReader.
* IDictionary<TKey, TValue> – названия колонок всегда Key и Value. Тип колонок будет соответствовать типам TKey и TValue.
* IEnumerable<IDictionary<string, TValue>> – названия колонок – ключи словаря, тип колонок будет соответствовать типу TValue.
* IEnumerable, IEnumerable<T> – по умолчанию в качестве названий столбцов выступают имена публичных полей и свойств объекта, который содержится в последовательности, имеющие примитивные типы данных, тип перечисления или типы Decimal, String, DateTime и Guid. Поведение по умолчанию можно переопределить с помощью специальных атрибутов, которые будут описаны ниже. Тип колонок будет соответствовать типу поля или свойства.
* Объект любого другого типа – аналогично, как и для типов IEnumerable и IEnumerable<T>.

Если в качестве источника данных выступают последовательности IEnumerable или IEnumerable<T> (или возможно одиночный объект), то, как уже упоминалось выше, в качестве столбцов по умолчанию принимаются публичные свойства и поля объекта, содержащегося в последовательности, которые имеют определённые типы данных. Это поведение можно уточнять с помощью следующих атрибутов:

* NoExcelColumn – помеченное данным атрибутом поле или свойство перестаёт считаться колонкой и не выводится в динамической панели;
* ExcelColumn – данный атрибут позволяет уточнить некоторые параметры колонки в Excel и имеет следующие свойства:
* Caption (тип string) – по умолчанию именем колонки в Excel будет являться имя свойства или поля, для задания более подходящего имени можно использовать данное свойство;
* DisplayFormat (тип string) – позволяет задать формат вывода для данной колонки, это свойство актуально для числовых данных, а также для дат; для типа Decimal по умолчанию задан формат #,0.00;
* IgnoreDisplayFormat (тип bool) – этот атрибут указывает, что необходимо игнорировать формат, даже если он указан в свойстве DisplayFormat, это может быть полезно, например, для сброса формата по умолчанию для типа Decimal, если он вас не устраивает;
* AggregateFunction (тип AggregateFunction) – позволяет задать агрегатную функцию, которая будет применяться к данной колонке в шаблоне {Totals}, это свойство имеет тип перечисления, которое содержит все стандартные агрегатные функции; для типа Decimal по умолчанию задана агрегатная функция Sum;
* NoAggregate (тип bool) – указывает, что не нужно применять к данной колонке какую-либо агрегацию, даже если она указана в поле AggregateFunction, это может быть полезно, например, для сброса агрегации по умолчанию для типа Decimal, если она вам не нужна;
* Width (тип int) – позволяет задать ширину колонки в случае вертикальной панели или высоту строки в случае горизонтальной;
* AdjustToContent (тип bool) – указывает, что нужно подогнать ширину колонки под размер содержимого;
* Order (тип int) – задаёт порядок колонки, в котором она появится на листе, по умолчанию это поле у всех равно нулю, поэтому колонки появляются в порядке объявления полей и свойств в типе, причём сначала идут все поля, а потом свойства.

1. Шаблон {Numbers} предназначен для вывода номера колонки в случае вертикальной панели или номера строки в случае горизонтальной. По умолчанию номер начинается с единицы, но данный шаблон может иметь расширенный вид, где можно задать число, с которого начнётся нумерация, в таком случае он, например, примет вид {Numbers(5)}. Это будет означать, что нумерация колонок (строк) начнётся с пяти.
2. Шаблон {Data} в принципе не очень примечателен, он просто выводит данные из источника данных посредством панели данных согласно определённым ранее столбцам. Нужно заметить, что если даже в панели не определён шаблон {Headers}, то столбцы для вывода всё равно будут определены корректно.
3. Шаблон {Totals} предназначен для вывода итогов. По умолчанию итоги считаются только для типа Decimal с помощью агрегатной функции Sum. Задать агрегатные функции для каких-либо других полей можно через поле AggregateFunction атрибута ExcelColumn или посредством функций жизненного цикла динамической панели. Данные функции будут описаны подробнее при рассмотрении свойств панели.

Для данной панели актуальные следующие свойства:

1. DataSource (обязательное, тип – любой, уместный в данном контексте шаблон) – полностью аналогично одноимённому свойству для панели данных.
2. RenderPriority (тип int) – полностью аналогично одноимённому свойству для простой панели.
3. Type (тип enum) – полностью аналогично одноимённому свойству для панели данных.
4. ShifType (тип enum) – полностью аналогично одноимённому свойству для панели данных.
5. GroupBy (тип string) – полностью аналогично одноимённому свойству для панели данных.
6. BeforeRenderMethodName (тип string) – полностью аналогично одноимённому свойству для панели данных.
7. AfterRenderMethodName (тип string) – полностью аналогично одноимённому свойству для панели данных.
8. BeforeHeadersRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом заголовков колонок или строк. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где BeforeHeadersRender – имя метода, которое указывается в поле комментария для свойства BeforeHeadersRenderMethodName, args – параметр метода, имеющий тип DataSourceDynamicPanelBeforeRenderEventArgs, который содержит в себе следующие поля:

* Range (тип ClosedXML.Excel.IXLRange) – диапазон ячеек, которые охватывает шаблон заголовка колонок до рендеринга;
* IsCanceled (тип bool) – флаг, который позволяет отменить рендеринг заголовков колонок;
* Data (тип object) – источник данных для данной панели;
* Columns (тип IList<ExcelDynamicColumn>) – колонки данных, определённые исходя из источника данных, в свою очередь тип ExcelDynamicColumn имеет следующие поля:
* Name (тип string) – наименование колонки в источнике данных;
* Caption (тип string) – наименование колонки (строки), которое будет отображено в Excel;
* Width (тип double?) – ширина колонки (или высота строки в случае горизонтальной панели);
* DataType (тип Type) – тип данных для данной колонки;
* AggregateFunction (тип AggregateFunction) – агрегатная функция, которая будет применяться к данной колонке;
* DisplayFormat (тип string) – формат, который будет применяться к данной колонке;
* AdjustToContent (тип bool) – указывает, что нужно подогнать ширину колонки под размер содержимого.

То есть, в данном методе вы имеете возможность настроить какую-либо колонку данных, задав необходимые свойства. Можно даже добавить новую колонку в коллекцию или удалить существующую, но в этом случае нужно будет также скорректировать источник данных, что тоже возможно сделать в этом методе. Во все остальные методы жизненного цикла уже будет передаваться скорректированная коллекция столбцов, а также скорректированный источник данных.

1. AfterHeadersRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга заголовков колонок или строк. Сам метод должен размещаться в экземпляре класса отчёта, который передаётся конструктору класса DefaultReportGenerator либо производного от него. Сигнатура данного метода должна иметь следующий вид:



где AfterHeadersRender – имя метода, которое указывается в поле комментария для свойства AfterHeadersRenderMethodName, args – параметр метода, имеющий тип DataSourceDynamicPanelEventArgs, который содержит в себе все те же поля, что и тип DataSourceDynamicPanelBeforeRenderEventArgs, описанный ранее, кроме поля IsCanceled. В этом случае поле Range будет содержать диапазон ячеек, который охватывает заголовки колонок после рендеринга.

1. BeforeNumbersRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом номеров колонок или строк. Во всём остальном этот метод полностью аналогичен методу BeforeHeadersRenderMethodName.
2. AfterNumbersRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга номеров колонок или строк. Во всём остальном этот метод полностью аналогичен методу AfterHeadersRenderMethodName.
3. BeforeDataTemplatesRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом шаблонов элементов данных. Динамическая панель выводит данные посредством панели данных, поэтому сначала шаблон {Data} превращается в шаблоны элементов данных, а затем получившийся диапазон с этими шаблонами передаётся панели данных для рендеринга. Во всём остальном этот метод полностью аналогичен методу BeforeHeadersRenderMethodName.
4. AfterDataTemplatesRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга шаблонов элементов данных. Во всём остальном этот метод полностью аналогичен методу AfterHeadersRenderMethodName. Здесь можно изменить каким-либо образом сформированный шаблон элемента данных.
5. BeforeDataRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед выводом данных. Этот метод соответствует методу BeforeRenderMethodName панели данных.
6. AfterDataRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после вывода данных. Этот метод соответствует методу AfterRenderMethodName панели данных.
7. BeforeDataItemRenderMethodName (тип string) – этот метод соответствует одноимённому методу панели данных.
8. AfterDataItemRenderMethodName (тип string) – этот метод соответствует одноимённому методу панели данных.
9. BeforeTotalsTemplatesRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед рендерингом шаблонов агрегатных функций. Динамическая панель выводит итоги посредством панели итогов, поэтому сначала шаблон {Totals} превращается в шаблоны агрегатных функций, а затем получившийся диапазон с этими шаблонами передаётся панели итогов для рендеринга. Во всём остальном этот метод полностью аналогичен методу BeforeHeadersRenderMethodName.
10. AfterTotalsTemplatesRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после рендеринга шаблонов агрегатных функций. Во всём остальном этот метод полностью аналогичен методу AfterHeadersRenderMethodName. Здесь можно изменить каким-либо образом сформированный шаблон агрегатной функции, например, добавить функцию постагрегации.
11. BeforeTotalsRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно перед выводом итогов. Этот метод соответствует методу BeforeRenderMethodName панели итогов.
12. AfterTotalsRenderMethodName (тип string) – позволяет задать имя метода, который будет выполнен непосредственно после вывода итогов. Этот метод соответствует методу AfterRenderMethodName панели итогов.
13. ParentPanel (тип string) – аналогично одноимённому свойству для панели данных. При вкладывании динамической панели внутрь панели данных существуют некоторые ограничения:

* динамическую панель можно вкладывать в панель данных, только если они имеют одно и то же значение в свойстве Type (обе вертикальные или обе горизонтальные);
* из вложенной динамической панели нельзя обращаться к контексту родительской панели через указатель parent.

## Подключение и расширение

Подключить библиотеку довольно легко. Для этого предназначен класс DefaultReportGenerator, конструктор которого принимает единственный параметр – экземпляр отчёта типа object, который необходимо вывести в Excel. Для запуска рендеринга служит метод Render, имеющий следующую сигнатуру:



где reportTemplate – это книга Excel, размеченная необходимыми панелями и шаблонами, worksheets – массив листов, которые должны быть обработаны движком. Если массив пуст или равен null, то будут обработаны все листы. Возвращаемое значение – это готовый отчёт, где все панели и шаблоны заменены реальными данными.

Класс DefaultReportGenerator имеет следующие события:

1. BeforeReportRender – происходит непосредственно перед началом рендеринга всей книги Excel. Имеет тип EventHandler<ReportRenderEventArgs>, где ReportRenderEventArgs содержит единственное поле Workbook типа XLWorkbook – это шаблон книги Excel, который был передан методу Render первым параметром.
2. BeforeWorksheetRender – происходит непосредственно перед рендерингом каждого листа. Имеет тип EventHandler<WorksheetRenderEventArgs>, где WorksheetRenderEventArgs содержит единственное поле Worksheet типа IXLWorksheet.
3. AfterWorksheetRender – происходит непосредственно после рендеринга каждого листа. Имеет точно такой же тип, как и событие BeforeWorksheetRender.

Класс DefaultReportGenerator имеет возможности для расширения и настройки. Рассмотрим их подробнее.

* 1. **Расширение системных функций.** Класс DefaultReportGenerator имеет свойство SystemFunctionsType типа Type. По умолчанию данному свойству присвоен тип SystemFunctions. Для расширения системных функций необходимо создать свой класс, унаследованный от класса SystemFunctions, определить в нём необходимые вам функции, а затем присвоить тип данного класса свойству SystemFunctionsType объекта DefaultReportGenerator.
  2. **Расширение системных переменных.** Класс DefaultReportGenerator имеет свойство SystemVariableProvider одноимённого типа SystemVariableProvider. По умолчанию данному свойству присвоен экземпляр типа SystemVariableProvider, который содержит системные переменные, поставляемые библиотекой. Для расширения системных переменных необходимо создать свой класс, унаследованный от класса SystemVariableProvider, а затем присвоить экземпляр данного класса свойству SystemVariableProvider объекта DefaultReportGenerator.
  3. **Расширение или замена реализации ITypeProvider.** Класс DefaultReportGenerator имеет свойство TypeProvider типа ITypeProvider. По умолчанию данному свойству присвоен экземпляр класса DefaultTypeProvider. Интерфейс ITypeProvider предоставляет типы, указанные в шаблоне явно или неявно. Если тип в шаблоне не указан, например, {p:Name}, то DefaultTypeProvider возвратит тип экземпляра объекта, переданного в конструкторе класса DefaultReportGenerator, то есть свойство Name будет искаться именно в этом типе. Если же тип указан явно, например, {p:Company:Name}, то DefaultTypeProvider будет искать тип Company в исполняемой сборке. Если тип не будет найден, либо будет найдено более одного типа с данным именем, то будет выброшено соответствующее исключение. Явное указание типа в шаблоне можно дополнить пространством имён, например, {p:Reports.Common:Company:Name}, в таком случае тип Company будет искаться с учётом пространства имён Reports.Common. Всё это справедливо для любых шаблонов, где можно указывать тип.

По умолчанию все типы ищутся в исполняемой сборке, но это поведение можно изменить. Конструктор класса DefaultTypeProvider принимает два параметра:

* assemblies (тип ICollection<Assembly>) – это коллекция сборок, в которых следует искать типы; если коллекция пуста или равна null, то типы ищутся в исполняемой сборке, значение по умолчанию – null;
* defaultType (тип Type) – тип, который будет возвращаться, если в шаблоне отсутствует явное указание типа; значение по умолчанию – null; при создании экземпляра DefaultTypeProvider внутри класса DefaultReportGenerator данному параметру присваивается тип объекта, переданного в конструкторе.

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство TypeProvider, которое будет возвращать нужную реализацию. В качестве данной реализации может выступать экземпляр класса DefaultTypeProvider, созданный с нужными параметрами конструктора, а также вы можете расширить класс DefaultTypeProvider или реализовать интерфейс ITypeProvider с нуля. Данный интерфейс имеет единственный метод со следующей сигнатурой:



Метод принимает строковый шаблон типа и должен возвратить тип, соответствующий данному шаблону.

* 1. **Расширение или замена реализации IInstanceProvider.** Класс DefaultReportGenerator имеет свойство InstanceProvider типа IInstanceProvider. По умолчанию данному свойству присвоен экземпляр класса DefaultInstanceProvider. Интерфейс IInstanceProvider предоставляет экземпляры объектов для типов, указанных в шаблонах. Тип DefaultInstanceProvider имеет единственный параметр конструктора, в котором можно указать экземпляр объекта, который будет возвращаться, если в шаблоне тип не указан явно. При создании экземпляра DefaultInstanceProvider внутри класса DefaultReportGenerator данному параметру присваивается экземпляр объекта, переданного в конструкторе. Поэтому, когда в шаблоне тип не указан явно, то все свойства, методы и т.п. вызываются на объекте отчёта. Нужно учитывать, что для того чтобы DefaultInstanceProvider смог создать экземпляр объекта, тип должен иметь конструктор по умолчанию. Все объекты, предоставляемые классом DefaultInstanceProvider, являются Singleton-объектами, то есть, если в нескольких шаблонах указан один и тот же тип, то все обращения к экземплярным членам этого типа будут происходить на объекте, созданном при первом обращении.

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство InstanceProvider, которое будет возвращать нужную реализацию. В качестве данной реализации может выступать экземпляр класса DefaultInstanceProvider, созданный с нужными параметрами конструктора, а также вы можете расширить класс DefaultInstanceProvider или реализовать интерфейс IInstanceProvider с нуля. Данный интерфейс имеет два метода со следующей сигнатурой:



Оба метода должны возвращать экземпляр объекта указанного типа. Второй метод – это просто обобщённая версия первого. Например, реализацию по умолчанию можно заменить реализацией, которая будет получать экземпляры объектов посредством какого-либо IoC-контейнера.

* 1. **Расширение или замена реализации IPropertyValueProvider.** Класс DefaultReportGenerator имеет свойство PropertyValueProvider типа IPropertyValueProvider. По умолчанию данному свойству присвоен экземпляр класса DefaultPropertyValueProvider. Интерфейс IPropertyValueProvider предоставляет значения для шаблона свойств. Тип DefaultPropertyValueProvider принимает в конструкторе два параметра:
* typeProvider (тип ITypeProvider) – экземпляр объекта, который будет предоставлять типы, указанные в шаблоне; в классе DefaultReportGenerator этому параметру передаётся значение свойства TypeProvider, описанного выше;
* instanceProvider (тип IInstanceProvider) – экземпляр объекта, который будет предоставлять экземпляры объектов для указанных в шаблоне типов; в классе DefaultReportGenerator этому параметру передаётся значение свойства InstanceProvider, описанного выше.

Реализация DefaultPropertyValueProvider поддерживает обращение к публичным полям и свойствам, причём как статическим, так и экземплярным. Также есть возможность склеивать обращения к полям и свойствам через «.».

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство PropertyValueProvider, которое будет возвращать нужную реализацию. В качестве данной реализации может выступать экземпляр класса DefaultPropertyValueProvider, созданный с нужными параметрами конструктора, а также вы можете расширить класс DefaultPropertyValueProvider или реализовать интерфейс IPropertyValueProvider с нуля. Данный интерфейс имеет единственный метод со следующей сигнатурой:



Данный метод принимает на вход шаблон свойства и должен возвратить значение этого свойства исходя из данного шаблона.

* 1. **Расширение или замена реализации IMethodCallValueProvider.** Класс DefaultReportGenerator имеет свойство MethodCallValueProvider типа IMethodCallValueProvider. По умолчанию данному свойству присвоен экземпляр класса DefaultMethodCallValueProvider. Интерфейс IMethodCallValueProvider предоставляет значения для шаблонов вызова методов. Тип DefaultMethodCallValueProvider принимает в конструкторе те же аргументы, что и тип DefaultPropertyValueProvider.

Реализация DefaultMethodCallValueProvider поддерживает вызов как статических, так и экземплярных публичных методов. Методу можно передавать сколько угодно параметров, причём в качестве параметров могут выступать как статические данные, зашитые внутри шаблона (строки, числа и т.п.), так и практически любые типы шаблонов, включая шаблоны вызова методов, причём вложенность вызовов не ограничена. Методы с переменным числом параметров (params) не поддерживаются. Перегрузка поддерживается частично, иногда для корректного определения перегруженного метода необходимо указывать в шаблоне типы статических параметров явно. Если вдруг не удастся однозначно определить перегруженный метод, то будет выброшено соответствующее исключение.

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство MethodCallValueProvider, которое будет возвращать нужную реализацию. В качестве данной реализации может выступать экземпляр класса DefaultMethodCallValueProvider, созданный с нужными параметрами конструктора, а также вы можете расширить класс DefaultMethodCallValueProvider или реализовать интерфейс IMethodCallValueProvider с нуля. Данный интерфейс имеет два метода:



Данный метод принимает на вход шаблон вызова метода, экземпляр обработчика шаблонов (будет описан ниже) и элемент данных, если шаблон находится в контексте данных. Возвращает результат вызова метода.



Этот метод отличается от предыдущего тем, что принимает ещё один параметр concreteType типа Type. В данном случае тип, в котором следует искать методы указан явно, в остальном он полностью аналогичен вышеописанному.

* 1. **Расширение или замена реализации IGenericDataItemValueProvider<HierarchicalDataItem>.** Класс DefaultReportGenerator имеет свойство DataItemValueProvider типа IGenericDataItemValueProvider<HierarchicalDataItem>. По умолчанию данному свойству присвоен экземпляр класса HierarchicalDataItemValueProvider. Интерфейс IGenericDataItemValueProvider<HierarchicalDataItem> предоставляет значения для шаблонов элементов данных. Класс HierarchicalDataItem представляет собой иерархический элемент данных. HierarchicalDataItemValueProvider не принимает в конструкторе никаких параметров.

Реализация HierarchicalDataItemValueProvider поддерживает вывод различных элементов данных (DataRow, KeyValuePair<TKey, TValue>, объект любого типа и т.п.). При обращении к элементу данных можно склеивать обращения к полям и свойствам через «.». Также поддерживается иерархичность данных за счёт указателя parent в шаблоне.

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство DataItemValueProvider, которое будет возвращать нужную реализацию. В качестве данной реализации вы можете расширить класс HierarchicalDataItemValueProvider или реализовать интерфейс IGenericDataItemValueProvider<HierarchicalDataItem> с нуля. Данный интерфейс имеет два метода:



Данный метод принимает на вход шаблон элемента данных и сам элемент данных. Возвращает значение из элемента данных согласно шаблону.



Обобщённая версия первого метода.

* 1. **Расширение или замена реализации ITemplateProcessor.** Класс DefaultReportGenerator имеет свойство TemplateProcessor типа ITemplateProcessor. По умолчанию данному свойству присвоен экземпляр класса DefaultTemplateProcessor. Интерфейс ITemplateProcessor отвечает за парсинг и обработку всех шаблонов. Тип DefaultTemplateProcessor принимает в конструкторе следующие аргументы:
* propertyValueProvider (тип IPropertyValueProvider) – экземпляр объекта, предоставляющий значения для шаблонов свойств; в классе DefaultReportGenerator этому параметру передаётся значение свойства PropertyValueProvider, описанного выше;
* systemVariableProvider (тип SystemVariableProvider) – экземпляр объекта, предоставляющий значения системных переменных; в классе DefaultReportGenerator этому параметру передаётся значение свойства SystemVariableProvider, описанного выше;
* methodCallValueProvider (тип IMethodCallValueProvider) – экземпляр объекта, предоставляющий значения для шаблонов вызова методов; в классе DefaultReportGenerator этому параметру передаётся значение свойства MethodCallValueProvider, описанного выше;
* dataItemValueProvider (тип IGenericDataItemValueProvider<HierarchicalDataItem>) – экземпляр объекта, предоставляющий значения для шаблонов элементов данных; в классе DefaultReportGenerator этому параметру передаётся значение свойства DataItemValueProvider, описанного выше.

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство TemplateProcessor, которое будет возвращать нужную реализацию. В качестве данной реализации может выступать экземпляр класса DefaultTemplateProcessor, созданный с нужными параметрами конструктора, а также вы можете расширить класс DefaultTemplateProcessor или реализовать интерфейс ITemplateProcessor с нуля. Данный интерфейс имеет следующие свойства только для чтения:

* LeftTemplateBorder (тип string) – левая граница шаблона, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «{»;
* RightTemplateBorder (тип string) – правая граница шаблона, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «}»;
* MemberLabelSeparator (тип string) – разделитель, отделяющий тип (метку) шаблона от непосредственно самого шаблона, не может иметь значение null или пустой строки, в классе DefaultTemplateProcessor это свойство имеет значение «:»;
* PropertyMemberLabel (тип string) – метка шаблона свойств, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «p»;
* MethodCallMemberLabel (тип string) – метка шаблона вызова методов, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «m»;
* DataItemMemberLabel (тип string) – метка шаблона элемента данных, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «di»;
* SystemVariableMemberLabel (тип string) – метка шаблона системной переменной, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «sv»;
* SystemFunctionMemberLabel (тип string) – метка шаблона системной функции, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «sf»;
* HorizontalPageBreakLabel (тип string) – метка шаблона горизонтального разрыва страницы, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «HorizPageBreak»;
* VerticalPageBreakLabel (тип string) – метка шаблона вертикального разрыва страницы, не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство имеет значение «VertPageBreak»;
* DataItemSelfTemplate (тип string) – шаблон, который используется, для возврата самого элемента данных, а ни какого-либо поля или свойства из него (полезно, когда элемент данных имеет простой тип), не может иметь значение null, пустой или пробельной строки, в классе DefaultTemplateProcessor это свойство возвращает значение свойства DataItemMemberLabel, описанного выше, поэтому по умолчанию шаблон для возврата самого элемента данных имеет вид {di:di}.

Также данный интерфейс имеет единственный метод со следующей сигнатурой:



Данный метод принимает на вход шаблон и элемент данных, если шаблон находится в контексте данных. Возвращает значение шаблона.

* 1. **Настройка параметров парсинга панелей.** Класс DefaultReportGenerator имеет свойство PanelParsingSettings одноимённого типа PanelParsingSettings. Класс PanelParsingSettings имеет следующие поля для настройки:
* PanelPrefixSeparator (тип string) – разделитель между префиксом, обозначающим тип панели, и названием панели, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «\_»;
* SimplePanelPrefix (тип string) – префикс для простой панели, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «s»;
* DataSourcePanelPrefix (тип string) – префикс для панели данных, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «d»;
* DynamicDataSourcePanelPrefix (тип string) – префикс для динамической панели, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «dyn»;
* TotalsPanelPrefix (тип string) – префикс для панели итогов, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «t»;
* PanelPropertiesSeparators (тип string[]) – массив разделителей между свойствами панели, не может иметь значение null или быть пустым, по умолчанию имеет значение «["\n", "\t", ";"]», то есть свойства могут разделяться либо символом новой строки, либо символом табуляции, либо точкой с запятой;
* PanelPropertyNameValueSeparator (тип string) – разделитель между именем и значением свойства панели, не может иметь значение null, пустой или пробельной строки, по умолчанию имеет значение «=».

Чтобы изменить поведение по умолчанию, необходимо создать класс, унаследованный от класса DefaultReportGenerator и переопределить свойство PanelParsingSettings, которое будет возвращать экземпляр класса PanelParsingSettings с нужными настройками.