**Руководство по использованию библиотеки ExcelReportGenerator**

# Введение

Библиотека ExcelReportGenerator позволяет выводить данные в Microsoft Excel, размечая листы в нём определённым образом. В основе библиотеки лежат шаблоны и панели. 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, то будет выброшено исключение InvalidOperationException. 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 случай 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.

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), ключ 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}

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.

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.

# 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. 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. 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.
3. 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.

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. 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:

* DataTable – элементом данных выступает объект DataRow, в шаблоне указывается имя колонки.
* 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.
* 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.

* 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 «.»).
* 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.
* 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 «.»).
* 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) – имеет точно такое же значение, как и 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 перезатирает 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. BeforeRenderMethodName (тип string) – имеет точно такое же значение, как и 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) – имеет точно такое же значение, как и 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.

Рассмотри 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.

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 по сути не ограничена, то можно получить доступ к любому родителю в иерархии, тогда указатель parent нужно будет повторить нужно количество раз.