**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, Char, 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:

1. object GetDictVal(object dictionary, object key)

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.

1. object TryGetDictVal(object dictionary, object key)

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

1. object GetByIndex(object list, int index)

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.

1. object TryGetByIndex(object list, int index)

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

1. string Format(object input, string format, IFormatProvider provider = 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.