Лайфхак «Делаем первый компонент»

В данной статье мы рассмотрим пошаговую разработку компонента в Loginom на примере компонента **Генератор календаря** (см. Рисунок 1), который доступен в открытой библиотеке Loginom Main Library.

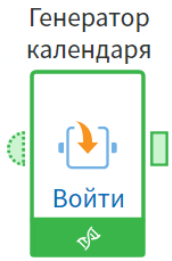

Рисунок 1 — Компонент "Генератор календаря"

1. Проектирование внешней оболочки

На первом уровне в рабочей области любой компонент должен представлять собой один узел. Для этого используется специальный компонент из группы *Управление* – **Подмодель**, который позволяет хранить внутри себя сценарий или какую-то его часть. Добавим его в рабочую область.

Перед проектированием структуры точнее сформулируем задачу: нам требуется компонент, который будет генерировать последовательный временной ряд с шагом в один день между заданными датами. По сути, это – календарь на заданный период, назовем подмодель **Генератор календаря**.

Исходя из задачи, мы понимаем, что на вход компонента должны подаваться два параметра: первая и последняя даты требуемого временного ряда, то есть у нашего компонента будет только один входной порт с типом **Переменные**. Зададим входному порту конкретную метку – **Даты**.

Выходом компонента будет сгенерированный временной ряд, то есть набор данных, значит, нам понадобится один выходной порт типа **Таблица**.

В настройках входного порта (см. Рисунок 2) создадим требуемые переменные с типом **Дата/Время**:

1) В поле **Имя** для первой переменной укажем *PeriodStart*, в поле **Метка** – *Период начальный*. Зададим значение по умолчанию – *01.10.2015 00:00*.
2) Для второй переменной зададим имя *PeriodFinish*, метку – *Период конечный*, значение по умолчанию – *23.05.2017 00:00*.

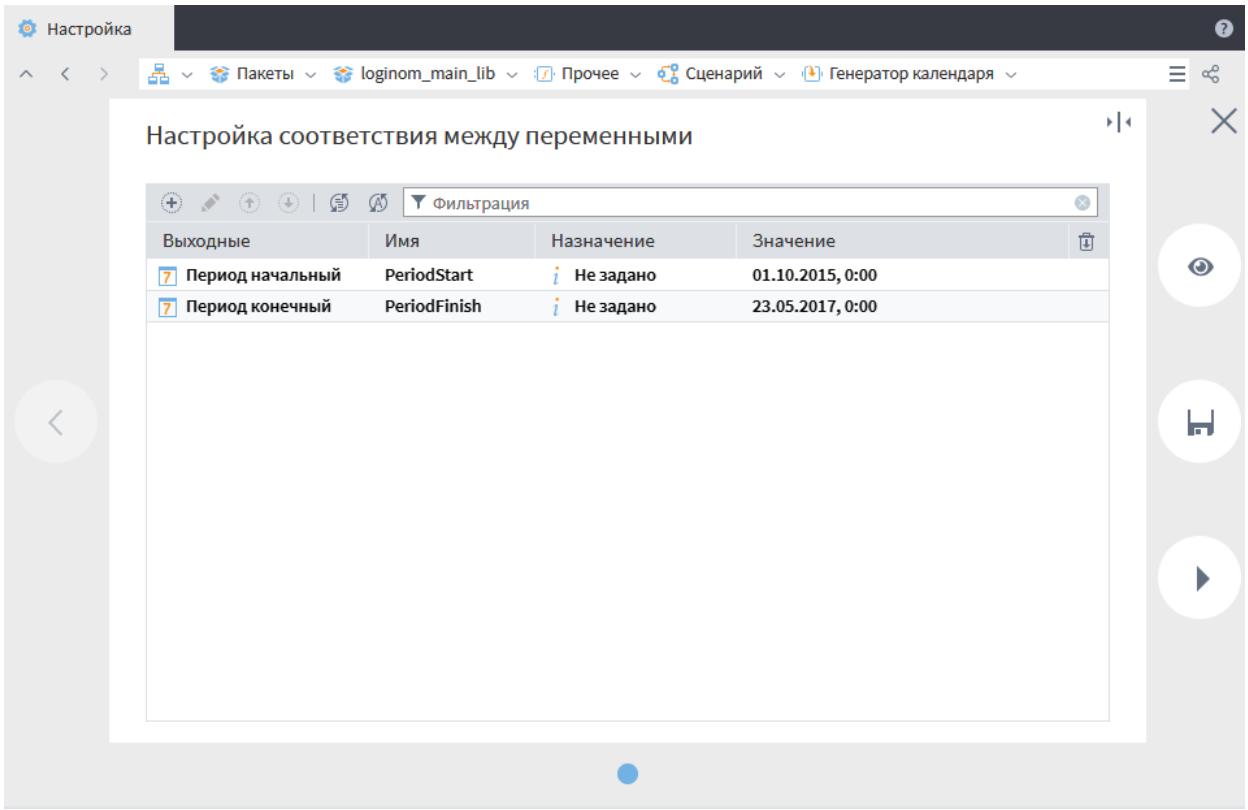

Рисунок 2 — Настройки входного порта

Поля на выходе компонента можно задать заранее или получить после создания сценария. Если в компоненте производится много вычислений и создается много промежуточных полей, лучше создать выходные поля сразу. Это позволит не искать, какие поля нужно оставить, а какие – удалить после завершения сценария. У нас на выходе будет всего одно поле, поэтому настраивать его заранее не обязательно.

После того, как мы определились со входами и выходами, переходим к проектированию внутренней структуры компонента.

2. Разработка сценария

Разберем, какие компоненты понадобятся для решения нашей задачи с учетом текущих возможностей и ограничений Loginom 6.2.3.

Для добавления дня к заданной дате можно воспользоваться функцией *AddDay()* компонента **Калькулятор (переменные)**. Но таким способом мы получим только следующую дату после заданной, тогда как необходимо, чтобы они генерировались до тех пор, пока не будет достигнута дата, указанная в переменной **Период конечный**.

Для динамической генерации дат нам понадобится *цикл с постусловием*, который можно реализовать с помощью стандартного компонента Loginom **Цикл**.

Для перевода наших переменных в табличный набор данных и обхода ряда ограничений платформы нам также понадобятся компоненты **Подмодель**, **Переменные в таблицу**, **Объединение** и **Фильтр строк**.

Подготовка цикла

Войдем в подмодель **Генератор календаря**. Чтобы воспользоваться циклом, сначала мы должны задать для него вычисляемое выражение с приращением и условием.

Начнем с добавления в область построения сценария компонента **Калькулятор (переменные)**. Подадим на вход узла переменные входного порта **Даты** и зайдем в настройки узла. У выражения по умолчанию изменим имя на *Calendar*, метку на *Календарь* и укажем тип выражения – *Дата/Время*. В области кода выражения зададим формулу *AddDay(PeriodStart,1)*, где *PeriodStart* – начальная дата, 1 – количество дней, на которое будет выполнено приращение к значению переменной, и сохраним настройки узла.

Чтобы в дальнейшем даты генерировались в столбце, необходимо превратить нашу новую переменную в поле набора данных. Воспользуемся

компонентом **Переменные в таблицу**, на вход которого подадим данные с выхода узла **Калькулятор (переменные)**. При настройках по умолчанию все три наши переменные становятся столбцами, но в дальнейшем нам понадобится только поле **Календарь**, поэтому в выходном порте узла **Переменные в таблицу** удалим выходные поля **Период начальный** и **Период конечный**.

Итак, мы получили дату со сдвигом от начальной на один день. Теперь нам необходимо поместить два созданных узла – **Калькулятор (переменные)** и **Переменные в таблицу** – внутрь цикла для динамической генерации дат.

Цикл может быть настроен только на один конкретный узел, поэтому выделим наши два узла и нажмем кнопку **Свернуть узлы в подмодель** на панели инструментов. Входной порт переменных создался автоматически, но выходных портов у подмодели нет, поэтому перейдем в ее настройки и добавим на выходы порты с типами *Таблица* и *Переменные*. Таблица понадобится нам для формирования набора данных, а переменные – для управления циклом. Переименуем все входы и выходы: метку входного порта обозначим как *Переменные*, а выходные порты как *Таблица* и *Переменные* соответственно. На следующем шаге назовем саму подмодель **Календарь** и сохраним настройки.

Войдем в подмодель (см. Рисунок 3) и подадим выходные данные с узла **Переменные в таблицу** на порт *Таблица*, а выходные данные с узла **Калькулятор** (переменные) на порт *Переменные*. В настройках порта переменных уберем **Период начальный** и **Период конечный** – для цикла они не понадобятся – и выйдем из подмодели.

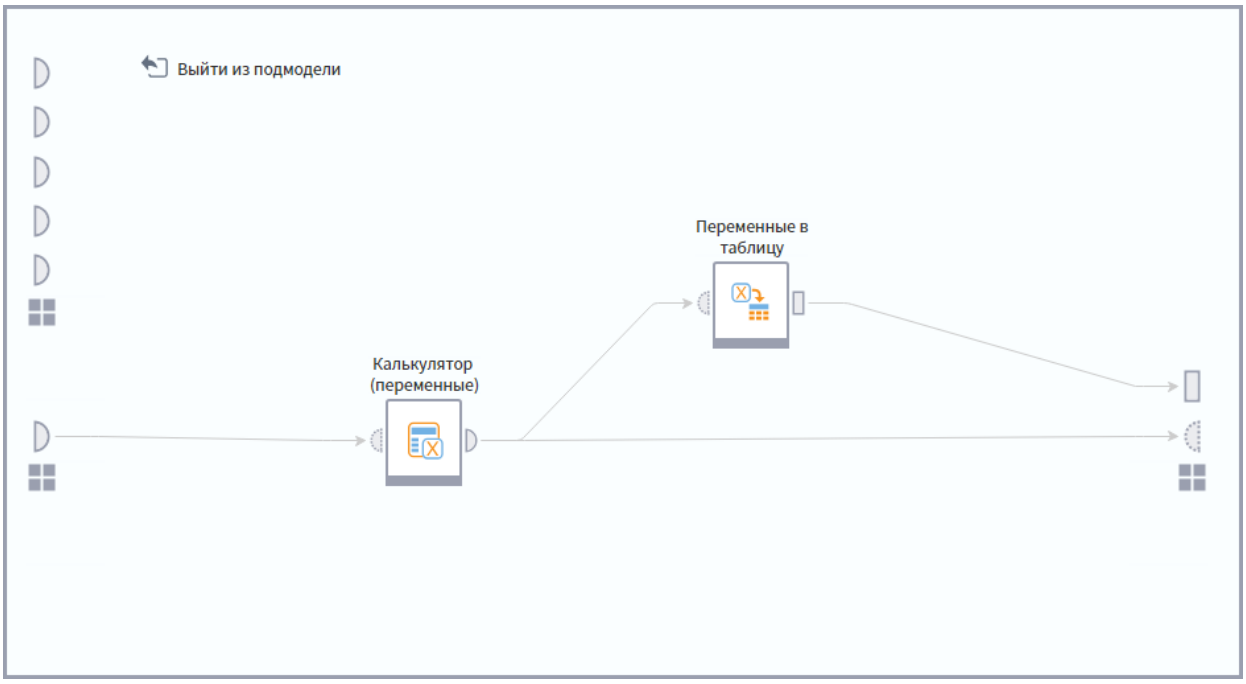

Рисунок 3 — Подмодель "Календарь"

Добавим в область построения компонент **Цикл**, в качестве узла цикла выберем подмодель **Календарь**. На следующем шаге (см. Рисунок 4) выберем *Цикл с постусловием*, в поле **Переменная** укажем *Переменные.Календарь*, условие завершения — “=”. Далее нам необходимо указать значение, по достижении которого цикл должен завершиться. Нужное значение хранится в переменной **Период конечный**, но на данный момент Loginom не позволяет задать для цикла управляющие переменные, и мы не можем ее использовать. Если мы укажем значение этой переменной в цикле вручную, то каждый раз при изменении входных данных пользователю придется заходить внутрь компонента и менять условие цикла. Однако компоненты должны проектироваться таким образом, чтобы все параметры задавались «снаружи», пользователь не должен менять что-либо внутри подмодели. Поэтому воспользуемся обходным путем – зададим в качестве значения дату, которая на несколько лет больше текущей, – *01.06.2021 00:00*.

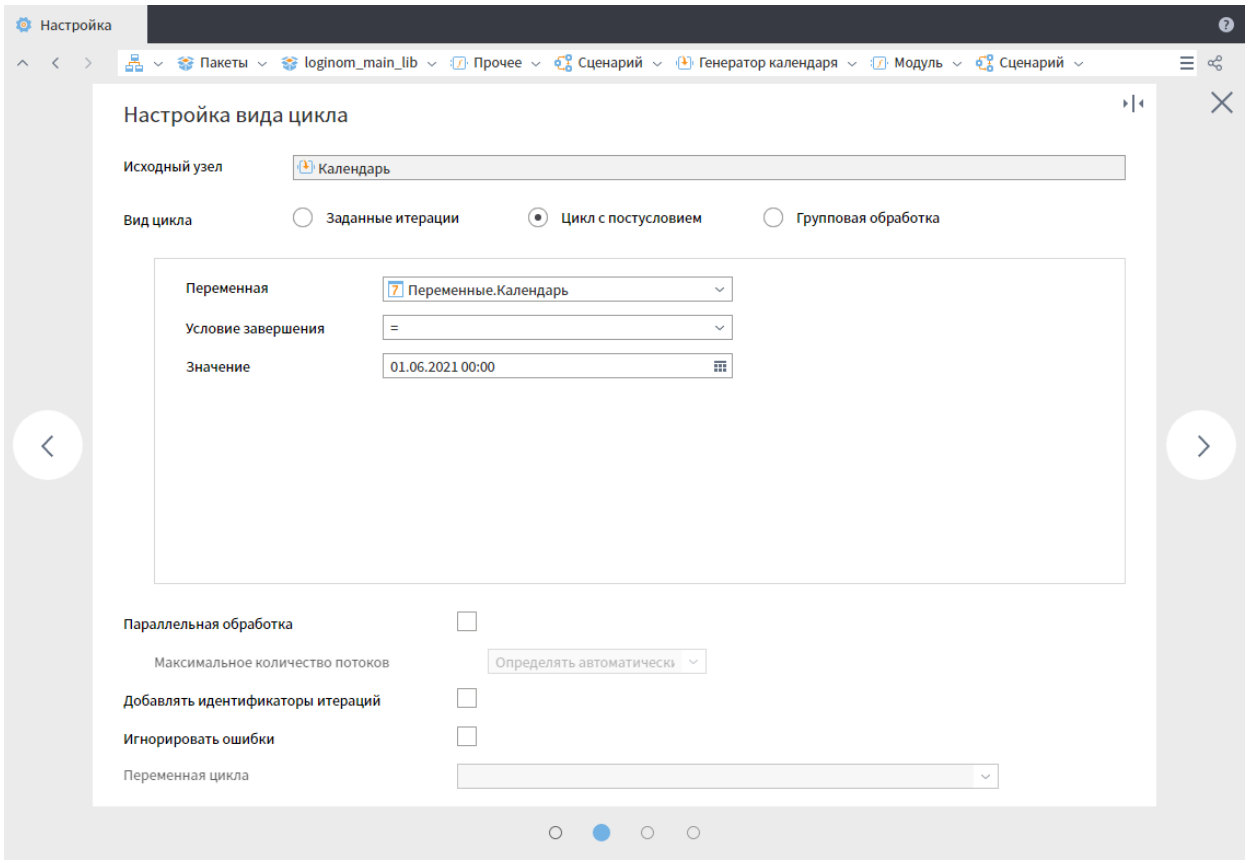

Рисунок 4 — Настройки цикла

На следующем шаге связываем выходную переменную **Переменные.Календарь** с входной переменной **Переменные.Период начальный**, далее задаем циклу метку – *Генератор календаря* – и сохраняем настройки. На вход генератора передаем данные с порта **Даты** нашей внешней подмодели.

Последующая обработка данных

После запуска цикла мы получаем последовательность дат, но она не удовлетворяет нашим изначальным требованиям: во-первых, отсутствует начальная дата, временной ряд начинается с даты *Период начальный + 1 день*; во-вторых, последовательность сгенерирована до указанного условия завершения – *01.06.2021 00:00*, а эта дата значительно больше заданной в переменной *Период конечный*.

Для того, чтобы первой датой нашего календаря была дата, указанная в переменной *Период начальный*, воспользуемся компонентами **Переменные в таблицу** и **Объединение**.

Добавим в сценарий узел **Переменные в таблицу**, подадим на его вход данные с входного порта **Даты** и изменим метку узла на *Период мин. в таблицу*. После этого зайдем в настройки выходного порта данного узла, уберем столбец **Период конечный**, а для поля **Период начальный** изменим имя на *Calendar* и метку на *Календарь*. Теперь наша начальная дата также содержится в столбце.

Добавим первую дату к сгенерированному ряду. Для этого воспользуемся компонентом **Объединение**, на первый порт которого – *Главная таблица* – подаём данные с узла **Период мин. в таблицу**, а на второй – *Присоединяемая таблица* – с узла **Генератор календаря**. В настройках узла **Объединение** поставим галочку для присоединяемой таблицы и сохраним изменения.

Теперь календарь начинается с правильной даты, осталось добиться того, чтобы завершался он также нужным значением. Для того, чтобы убрать лишние даты, воспользуемся компонентом **Фильтр строк**.

Добавим компонент в область построения и подадим на вход данные с узла **Объединение**. Нажмем правой кнопкой мыши на **Фильтр строк** и выберем опцию **Показать порт управляющих переменных**. На появившийся порт подадим данные с входного порта **Даты**.

В настройках узла **Фильтр строк** добавим новый фильтр: в параметре **Поле** выберем *Период начальный*, в условии — “<=”, а в качестве значения для сравнения укажем переменную *Период конечный (23.05.2017, 0:00)*. Сохраним настройки и зададим узлу метку *Меньше максимального*.

Передадим данные с порта **Соответствуют условию** нашего фильтра на выходной порт внешней подмодели **Набор данных** (см. Рисунок 5).

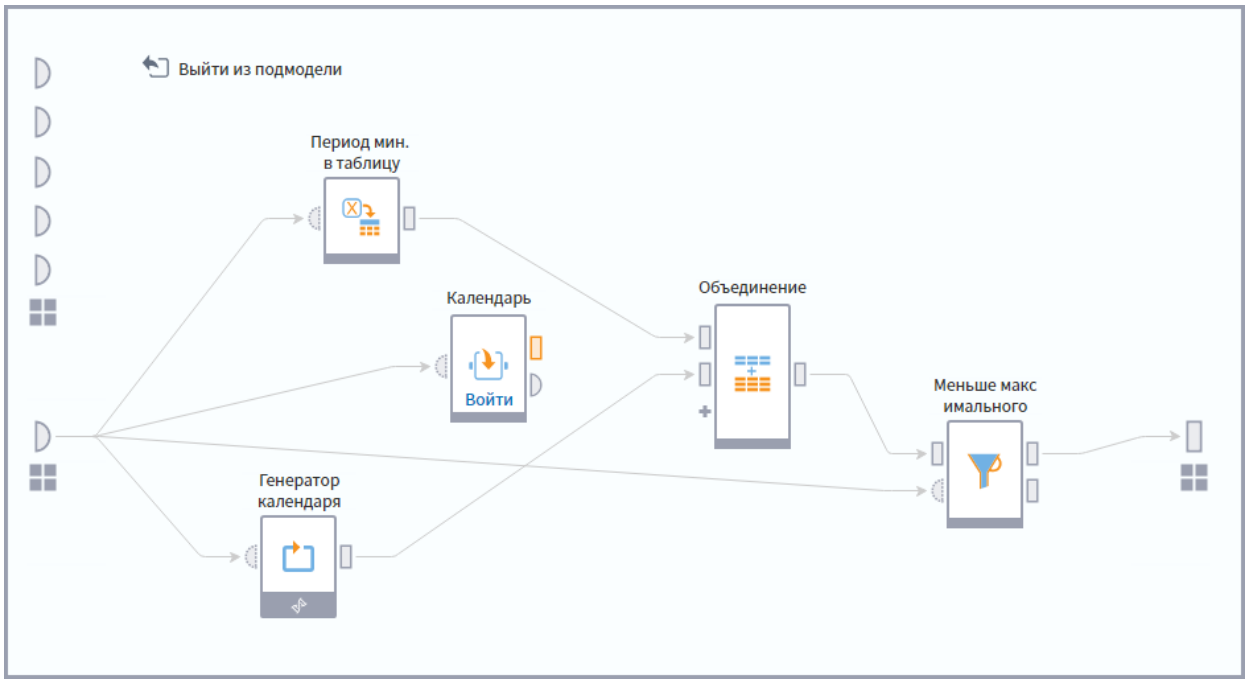

Рисунок 5 — Подмодель "Генератор календаря"

Выйдем из подмодели и активируем обработку. На выходе получим сгенерированный календарь, который соответствует нашим изначальным требованиям (см. Рисунок 6).

Генератор календаря • Набор данных • Быстрый просмотр данных

#	Календарь
1	01.10.2015, 0:00
2	02.10.2015, 0:00
3	03.10.2015, 0:00
4	04.10.2015, 0:00
5	05.10.2015, 0:00
6	06.10.2015, 0:00
7	07.10.2015, 0:00
8	08.10.2015, 0:00
9	09.10.2015, 0:00
10	10.10.2015, 0:00
11	11.10.2015, 0:00
12	12.10.2015, 0:00
13	13.10.2015, 0:00
14	14.10.2015, 0:00
601	15.10.2015, 0:00

Закрыть

Рисунок 6 — Сгенерированный набор данных

Теперь необходимо сделать так, чтобы наш узел был доступен в любом пакете.Щелкнем правой кнопкой мыши по подмодели **Генератор**

календаря, выберем пункт меню **Настроить модификатор доступа** и установим переключатель на вариант *Открытый (доступен во всех пакетах)*. Теперь мы можем использовать компонент в любом пакете с помощью стандартного компонента Loginom **Выполнение узла** при добавлении ссылки на данный пакет в любом другом пакете.

Так как библиотека Loginom Main Library является открытой, на базе узла **Генератор календаря** нами был также создан **производный компонент**. Производные компоненты с открытой областью видимости (аналог модификатора доступа для узла) доступны на слайд-панели на одноименной вкладке при добавлении ссылки на пакет с компонентами в любом другом пакете. При выполнении заданий Хакатона создавать производные компоненты не требуется.

Подробнее с работой через **Выполнение узла** и использованием производных компонентов можно познакомиться в курсе **Основы работы в Loginom II** в занятии **Занятие 5. Внешние компоненты и библиотеки**.

3. Резюме

В процессе разбора **Генератора календаря** мы упоминали ряд принципов, которые являются общими и могут быть использованы при проектировании любого компонента. Сформулируем их.

1. Название компонента должно отражать реализуемую функцию и состоять не более, чем из трех слов (не считая предлогов). Если компонентов несколько, их имена должны быть уникальными.
2. Если в сценарии используется несколько наборов данных, у компонента будет несколько входных портов, которым нужно давать осмысленные имена. Если же имеется не более, чем по одному порту для набора данных и переменных, можно именовать их «Входной набор данных» и «Переменные».
3. Если на выходе один выходной порт, рекомендуется называть его «Набор данных». Если портов несколько, основной можно именовать также, остальные выходы должны иметь осмысленные имена. Если данные с выхода одного компонента нужно подать на вход другого, имена связываемых портов должны совпадать.

4. Если на вход необходимо подать некие параметры обработки, они должны задаваться с помощью переменных непосредственно во входном порте переменных компонента.
5. Поля и переменные, которые участвуют в обработке, должны быть жестко заданы во входных портах компонента. Если компонент может обрабатывать несколько сущностей, скажем, считать некие метрики и по товарам, и по клиентам, в зависимости от того, что подано на вход, имена и метки полей должны быть осмысленными, но абстрактными, то есть называться не «Товар» или «Клиент», а «Объект».
6. Если на выходе подмодели присутствуют один или несколько общих для всего набора показателей, такие показатели следует выводить в переменных.
7. Чтобы сценарий был читаемым, и при необходимости можно было легко внести изменения, количество узлов внутри подмодели должно без труда помещаться на 2/3 экрана монитора. Если не помещаются, стоит разбить сценарий на несколько частей, каждую из которых «упаковать» в собственную подмодель.
8. Имена узлов, портов и полей, существующих внутри компонента, должны соответствовать принципам именования самого компонента, его портов и входных полей.