Приемы работы с универсальными коллекциями

Коллекция значений – это некий контейнер, в котором может содержаться обычно любое количество элементов. При этом каких-либо жестких ограничений на тип данных зачастую не накладывается. В универсальную коллекцию можно добавлять значения. Все значения в коллекции можно обойти.

Используются эти коллекции, в основном, для какой-либо обработки в алгоритмах. Т.е. это некие динамические структуры, которые существуют на время работы алгоритма.

Важно понимать, что коллекции не хранятся в базе данных (о типе данных *Хранилище значений*, которое может сохранять практически любой тип данных, речь пока не идет).

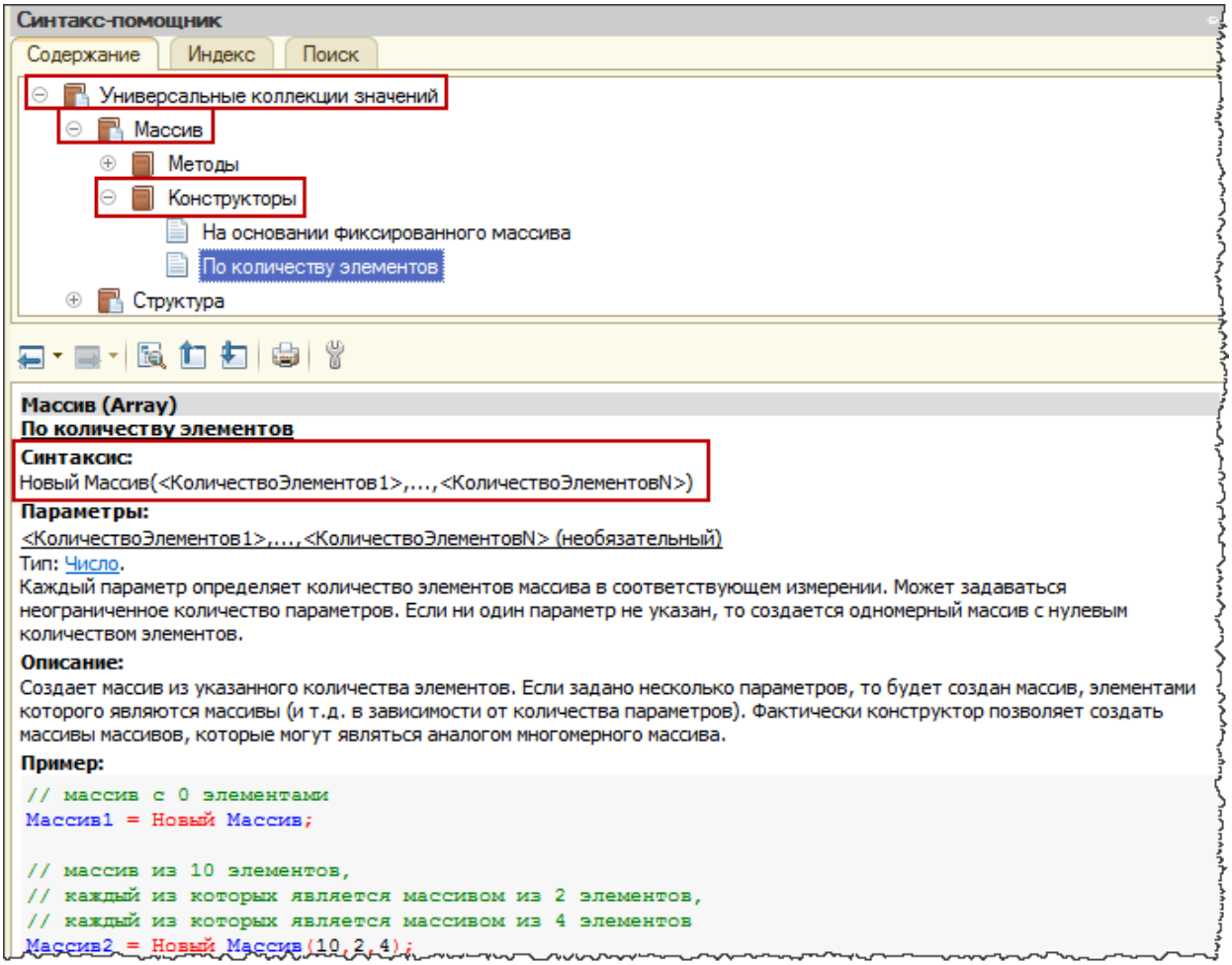
Существуют различные виды универсальных коллекций: *Массив*, *Структура*, *Соответствие*, *Фиксированный массив*, *Таблица значений*, *Табличная часть* и т.д. Но у всех коллекций есть схожесть поведения.

Коллекция может создаваться в результате работы какой-либо функции (функция возвращает в качестве значения универсальную коллекцию). Можно получить новую коллекцию вручную, обратившись к конструктору и создав экземпляр класса.

Например: *НашМассив = Новый Массив;*

Конструкторы для многих универсальных коллекций являются параметризованными.

Так, в конструкторе для *Массива* можно указать количество элементов в соответствующих измерениях. Т.е. можно сразу же объявлять многомерные *Массивы*. Соответствующее описание конструктора есть в синтакс-помощнике.



Таким образом, используя параметры конструктора, можно сразу задать желаемое поведение данного объекта.

Но параметры являются необязательными, разработчик может их не задавать и в дальнейшем определить поведение *Массива* так, как считает нужным.

Почти любую универсальную коллекцию можно создать с помощью конструктора (исключением являются табличные части, которые выступают в качестве объектов конфигурации).

Для универсальных коллекций существуют такие общие понятия, как индекс и номер. Каждый элемент коллекции имеет индекс. При этом индекс начинается с нуля.

Для того, чтобы обратиться к элементу *Массива НашМассив*, можно использовать обращение по индексу, для этого индекс указывается в квадратных скобках.

Например, *НашМассив[3]*. Обратите внимание, в этом случае система возвращает элемент *Массива* с индексом 3, а по порядку это четвертый элемент *Массива*. Для некоторых коллекций существует также понятие номера строки. Номер строки начинается с единицы. Например, для табличной части есть такое свойство, как номер строки.

Важно иметь ввиду, что если мы знаем номер строки и хотим обратиться по индексу, то в качестве индекса следует использовать значение на единицу меньше номера строки.

Понятие номера строки существует далеко не у всех коллекций, а преимущественно у тех, которые могут отображаться в интерфейсе пользователя.

Для всех коллекций используется обход элементов коллекции. Обход возможен двумя способами: *циклом Для* и *циклом Для каждого из*.

```
НашМассив = Новый Массив;

// Использование цикла Для
КоличествоВМассиве = НашМассив.Количество();
Для Индекс=0 По КоличествоВМассиве-1 Цикл

    Сообщить(НашМассив[Индекс]);

КонецЦикла;

// Использование цикла Для каждого
Для каждого ЭлементМассива Из НашМассив Цикл

    Сообщить(ЭлементМассива);

КонецЦикла;
```

Для большинства универсальных коллекций применимы методы: *Количество*, *Индекс*, *Добавить*, *Вставить*, *Удалить* и *Найти*.

Количество – это функция, которая возвращает количество элементов коллекции. Она может использоваться перед *циклом Для*, как представлено на рисунке.

Метод *Индекс* существует не у всех коллекций, а только у тех, на элементы которой можно сослаться. В качестве примера можно привести *ТаблицуЗначений*.

ТаблицаЗначений – это определенная коллекция строк, в строках могут содержаться разные колонки с разными типами значений.

Каждая строка представляет собой самостоятельную сущность. На нее можно получить ссылку, через эту строку можно обращаться к значениям колонок в данной строке.

Метод *Индекс* позволяет определить, какой индекс соответствует данной строке (т.е. текущую позицию строки в таблице). Значения индекса начинаются с нуля.

Методы добавления новых значений в данную коллекцию существуют практически у любой универсальной коллекции. На рисунке представлено, как заполнить *Массив* значениями от 0 до 10 двумя способами.

```
НашМассив = Новый Массив;

// Первый способ
Для Число=0 По 10 Цикл

    НашМассив.Добавить(Число);

КонецЦикла;

// Второй способ
Для Число=0 По 10 Цикл

    НашМассив.Вставить(0,Число);

КонецЦикла;
```

Для того, чтобы добавить элемент в *Массив* мы можем использовать метод *Добавить*, в скобках указать добавляемое значение. При этом значение будет добавляться в конец списка, т.е. *Массив* будет постоянно увеличиваться за счет последней позиции.

Другой метод, который позволяет добавлять значения в коллекцию – метод *Вставить*. Он отличается от метода *Добавить* тем, что можно указать, в какое место нужно вставить добавляемый элемент.

Синтаксис: *Вставить (<Индекс>,<Значение>)*

Первым параметром указывается индекс, в который будет вставлено новое значение. Т.е. мы, например, можем указать, что каждое значение нужно вставлять в начало списка (второй способ на рисунке выше).

Для удаления элементов из коллекции используется метод *Удалить*. В методе *Удалить* указывается по индексу, какой элемент мы будем удалять.

Синтаксис: *Удалить(<Индекс>)*

Пример использования: *НашМассив.Удалить(5);*

Следует отметить, что для тех коллекций, где строки представляют самостоятельную сущность (например, для *ТаблицыЗначений*), мы также можем использовать метод получения индекса для того, чтобы потом удалить данную строку.

Практически у всех коллекций существует метод поиска значения - *Найти*. В метод передается то значение, которое хотим найти. В некоторых коллекциях можно поставить какие-либо ограничения.

Например, в *ТаблицеЗначений* можно указать те строки, те колонки, в которых нужно осуществлять поиск. Если значение найдено, то данный метод возвращает индекс или определенную строку. Если значение не найдено, возвращается значение типа *Неопределено*. Применительно к *Массиву* возвращается *Индекс*, либо значение *Неопределено*.

Пример использования: *НашаПеременная = НашМассив.Найти(8);*

Универсальные коллекции можно очень быстро очищать, т.е. удалить абсолютно все элементы. Для этого используется метод *Очистить()*, который удаляет элементы *Массива*, строки *ТаблицыЗначений*, либо данные других коллекций.

Дополнительные методы для *Массива*

Метод *ВГраница()* возвращает количество элементов минус один. Т.е. если мы используем *цикл Для*, то вместо метода Количество можем сразу использовать метод *Граница()*.

В частности, переменную *КоличествоВМассиве* можно было определить иначе:

КоличествоВМассиве = НашМассив.ВГраница();

Тогда при описании самого цикла отнимать от данной переменной единицу не следует.

Метод *Установить* позволяет присвоить значение элементу *Массива* по индексу.

Синтаксис: *Установить(<Индекс>,<Значение>)*

Пример: *НашМассив.Установить (2,8);*

Альтернативный вариант: *НашМассив[2] = 8;*

Можно для *Массива* использовать метод *Получить*, для того, чтобы прочитать значение по индексу, не обращаясь к использованию квадратных скобок.

Синтаксис: *Получить(<Индекс>)*

Пример: *НашаПеременная = НашМассив.Получить(2);*

Альтернативный вариант: *НашаПеременная = НашМассив[2];*

Универсальная коллекция *Структура*

Структура, также, как и *Массив* может иметь неограниченное количество элементов, но вот содержание элемента отличается от *Массива*. *Структура* представляет собой коллекцию, каждое значение которой состоит из пары. Первый элемент пары называется *Ключ*. Второй элемент пары – *Значение*.

Ключ – это строго строковый тип данных, который описывает значение. Например, *Ключу* «Код» может соответствовать значение 113; *Ключу* «Имя» значение «Вася». На само Значение ограничение типа данных не накладывается.

Структуру очень удобно использовать, если мы хотим создать некий список параметров. Если данная *Структура* называется *НашаСтруктура*, то обращаться к ее двум значениям мы будем следующим образом: *НашаСтруктура.Код* и *НашаСтруктура.Имя*.

Такое обращение гораздо удобнее, чем если бы мы все параметры определили в *Массив* и обращались к ним по индексу. *Структура* делает программный код читаемым, (понятным). *Структура* применяется достаточно часто, гораздо чаще, чем *Массив*. Она используется для описания некоторых параметров, которых зачастую существует достаточно большое количество во всех алгоритмах.

Кроме того, *Структура* применяется в том случае, если процедура и функция содержат большое количество передаваемых параметров. Тогда гораздо удобнее записать все параметры в *Структуру* и именно ее и передавать. Т.е. происходит “упаковка” параметров процедур и функций.

Отдельно следует отметить, что в качестве *Ключа* в *Структуре* может выступать не абсолютно любая строка. Накладываются определенные ограничения.

Ключ должен выступать в качестве идентификатора. Это означает, что в *Ключе* не должно быть пробелов и *Ключе* не может начинаться с цифры. Допустимо начало *Ключа* с буквы или знака подчеркивания. Таким образом, *Ключ* должен удовлетворять требованиям к созданию идентификаторов.

Отметим, чем еще *Сруктура* отличается от *Массива*. В *Структуре* есть метод *Вставить*, в *Массиве* есть два метода для вставки: *Вставить* (в определенную позицию) и *Добавить* (в конец списка). В *Массиве* все элементы являются упорядоченными.

Структура – это некое неупорядоченное множество. Именно поэтому для *Структуры* существует только метод вставки. Значение вставляется не на конкретную позицию, а в указанное множество. Для *Структуры* недопустимо обращение по индексу, как для других универсальных коллекций.

К элементам Структуры обращаются только по имени *Ключа*. Тем не менее, *цикл Для каждого из* работает и для *Структуры*, но опираться на порядок элементов *Структуры* не следует. *Структура* создается точно так же, как и другие универсальные коллекции с помощью использования конструктора *Новый,* указывая тип данных *Структура*.

Как и у *Массива* конструктор *Структуры* может иметь параметры. Т.е. возможно описать само содержание *Структуры*, используя конструктор. В отличие от *Массива*, где можно просто указать количество элементов для всех размерностей, в *Структуре* возможно задавать само содержание.

Например: *НашаСтруктура = Новый Структура (“Код,Имя”, 133, “Вася”);*

Через запятую перечисляются сначала имена *Ключей*, а потом, соответственно, в той же последовательности значения параметров. Для добавления в *Структуру* нового значения существует метод *Вставить*, который вставляет новую пару (*Ключ* и *Значение*).

Например: *НашаСтруктура.Вставить(“ЧленовСемьи”,3);*

Для *Структуры* характерен еще один метод, который используется достаточно часто. Это метод *Свойство*. С помощью данного метода можно понять, а есть ли в этой *Структуре* такой элемент, у которого *Ключ* имеет такое-то имя.

Если существует такой элемент, то система вернет значение *Истина*, в противном случае - *Ложь*. Например, выражение *НашаСтруктура.Свойство (“ЧленовСемьи”)* будет равно значению *Истина*. Этот метод применяется достаточно часто при анализе Структуры.

Как и для любой универсальной коллекции допустимо обращение к свойствам Структуры

по индексу. Но индекс для Структуры – это строковое значение.

Например: *Сообщить(НашаСтруктура[“ЧленовСемьи”]);*

Однако следует не забывать, что Структура – это не упорядоченное множество объектов, именно поэтому обращение по индексу 0, 1, 2 недопустимо.

Универсальная коллекция *Список значений*

СписокЗначений представляет собой линейный список элементов любого типа данных. Каждый элемент состоит из нескольких значений. Схематично список значений можно представить в виде списка с четырьмя колонками.

Первая колонка – *Отметка*. Она имеет булевский тип данных и позволяет пользователю либо ставить флажки, либо их снимать.

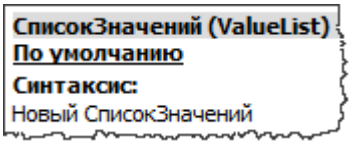
Другая колонка – это картинка, которая может каким-то образом визуально изображать данный элемент, т.е. ставить в соответствие данной строке какую-либо картинку.

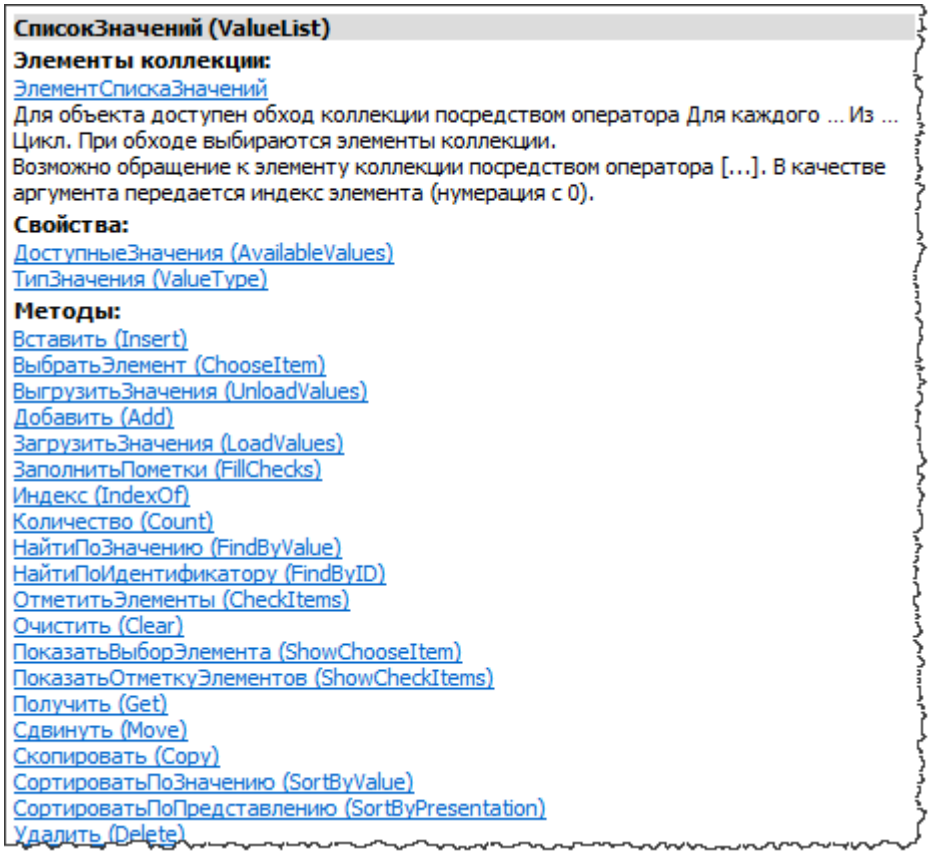
Третья колонка – само хранимое значение, т.е. это любой тип данных, причем в разных строках он может быть различным.

Четвертая колонка – это представление, т.е. это некое строковое описание данного значения. Представление будет выводиться пользователю, когда он будет просматривать данный элемент. При этом, если представление не задано, система будет пытаться сама получить представления для элемента, содержащегося в данной позиции.

СписокЗначений – это тот объект, с которым может визуально работать пользователь. Т.е. *СписокЗначений* можно вывести на форму. Пользователь может выполнять с ним какие-то действия. Кроме этого, *СписокЗначений* можно вывести независимо, используя методы, т.е. показать на экран в некоторой ветке алгоритма (за исключением серверного кода), чтобы пользователь выбрал какую-то строчку, либо проставил какие-либо галочки.

Найдем *СписокЗначений* в ситакс-помощнике. Конструктор *СпискаЗначений* не параметризованный (нельзя задать какие-то значения по умолчанию).





СписокЗначений (ValueList)

Элементы коллекции:

ЭлементСпискаЗначений

Для объекта доступен обход коллекции посредством оператора Для каждого ... Из ... Цикл. При обходе выбираются элементы коллекции.

Возможно обращение к элементу коллекции посредством оператора [...]. В качестве аргумента передается индекс элемента (нумерация с 0).

Свойства:

ДоступныеЗначения (AvailableValues)

ТипЗначения (ValueType)

Методы:

Вставить (Insert)

ВыбратьЭлемент (ChooseItem)

ВыгрузитьЗначения (UnloadValues)

Добавить (Add)

ЗагрузитьЗначения (LoadValues)

ЗаполнитьПометки (FillChecks)

Индекс (IndexOf)

Количество (Count)

НайтиПоЗначению (FindByValue)

НайтиПоИдентификатору (FindByID)

ОтметитьЭлементы (CheckItems)

Очистить (Clear)

ПоказатьВыборЭлемента (ShowChooseItem)

ПоказатьОтметкуЭлементов (ShowCheckItems)

Получить (Get)

Сдвинуть (Move)

Скопировать (Copy)

СортироватьПоЗначению (SortByValue)

СортироватьПоПредставлению (SortByPresentation)

Удалить (Delete)

Есть такие методы как:

- *Вставить(<Индекс>, <Значение>, <Представление>, <Пометка>, <Картинка>) ;*
- *Добавить(<Значение>, <Представление>, <Пометка>, <Картинка>);*
- *Количество();*
- *Индекс(<Элемент>).*

Есть и специальные методы, например, *ВыгрузитьЗначения()*. При этом создается *Массив*, в который копируется список значений. Например:

МассивЭлементов = СписокТиповЦен.ВызрузитьЗначения();

Существует и обратный метод:

СписокТиповЦен.ЗагрузитьЗначения(МассивЭлементов);

Существуют методы поиска:

НайтиПоЗначению(<ИскомоеЗначение>);
НайтиПоИдентификатору(<Идентификатор>).

Есть метод копирования:

КопияСписка = СписокТиповЦен.Скопировать();

Данный метод предназначен для того, чтобы сделать какую-то модификацию с копией.

Существуют методы:

СортироватьПоЗначению(<Направление>);

СортироватьПоПредставлению(<Направление>).

Методы *ВыбратьЭлемент(<Заголовок>,<Элемент>)* и *ОтметитьЭлементы (<Заголовок>)* вызывают модальное диалоговое окно, которое останавливает выполнение алгоритма, пока пользователь не закроет данное окно.

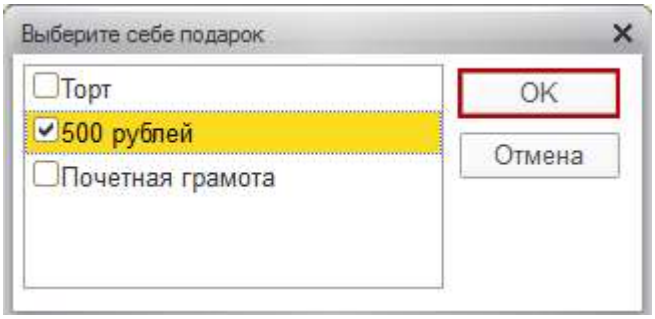
Для использования этих методов в свойствах конфигурации *Режим использования модальности* должен быть установлен на значение *Использовать*.

Пример кода, вызываемого из модуля Управляемого приложения:

```
Процедура ПриНачалеРаботыСистемы()

    УзнатьОподарке();

КонецПроцедуры
```

```
Процедура УзнатьОПодарке()

    // Создаем список значений
    СписокЗначений = Новый СписокЗначений;
    СписокЗначений.Добавить("Торт","Торт",Ложь,);
    СписокЗначений.Добавить("ДенежнаяПремия","500 рублей",Ложь,);
    СписокЗначений.Добавить("ПочетнаяГрамота","Почетная грамота",Ложь,);
    ВсегоВыбрано = 0;

// Цикл, пока не будет выбрано ровно одно значение.
Пока ВсегоВыбрано<>1 Цикл

    Подарок = СписокЗначений.ОтметитьЭлементы("Выберите себе подарок");
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Вы не сделали выбор.";

    Для каждого элемент Из СписокЗначений Цикл
        Если Элемент.Пометка Тогда
            ВсегоВыбрано = ВсегоВыбрано+1;
            Если  ВсегоВыбрано>1 Тогда
                Сообщение.Текст = "Можно выбрать только один подарок.";
                ВсегоВыбрано = 0;
                Прервать; //Если подарков больше 1 - цикл повторить.
            КонецЕсли;
            Подарок = Новый Структура("ВашПодарок", Элемент.Значение);
        КонецЕсли;
    КонецЦикла;

    Если ВсегоВыбрано = 1 Тогда // Выйдет сообщение и цикл прервется.
        Сообщение.Текст = "Ваш подарок - " + Подарок.ВашПодарок + ".";
    КонецЕсли;
    Сообщить(Сообщение.Текст);
КонецЦикла;

КонецПроцедуры
```

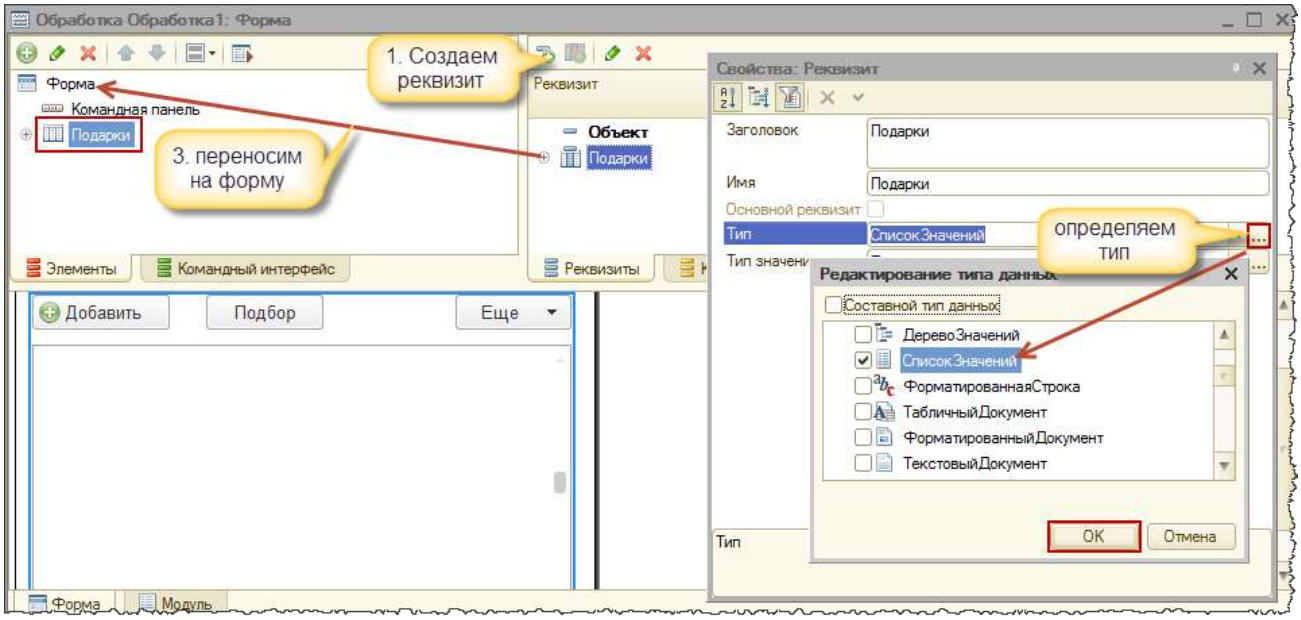
Отображение данного кода в пользовательском режиме (модальное диалоговое окно).



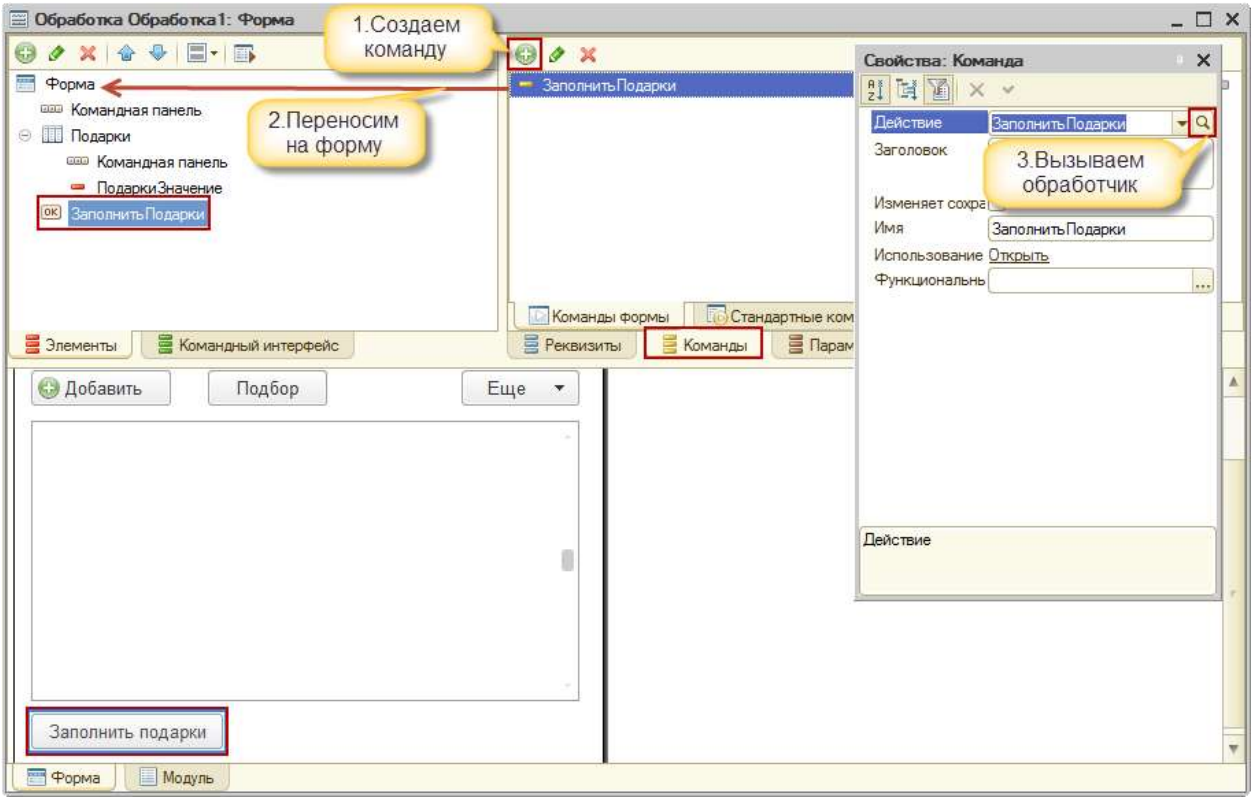
Бесплатные материалы проекта:
http://курсы-по-1с.рф/free/

Электронная почта: support@Kursy-po-1C.ru
Телефоны: **8-800 1000-613**, +7(495) 7777-096

Ниже *СписокЗначений* используется в качестве доступного типа данных для реквизита формы. Создаем для формы обработки новый реквизит, определяем для него тип *СписокЗначений* и отображаем его на форме.



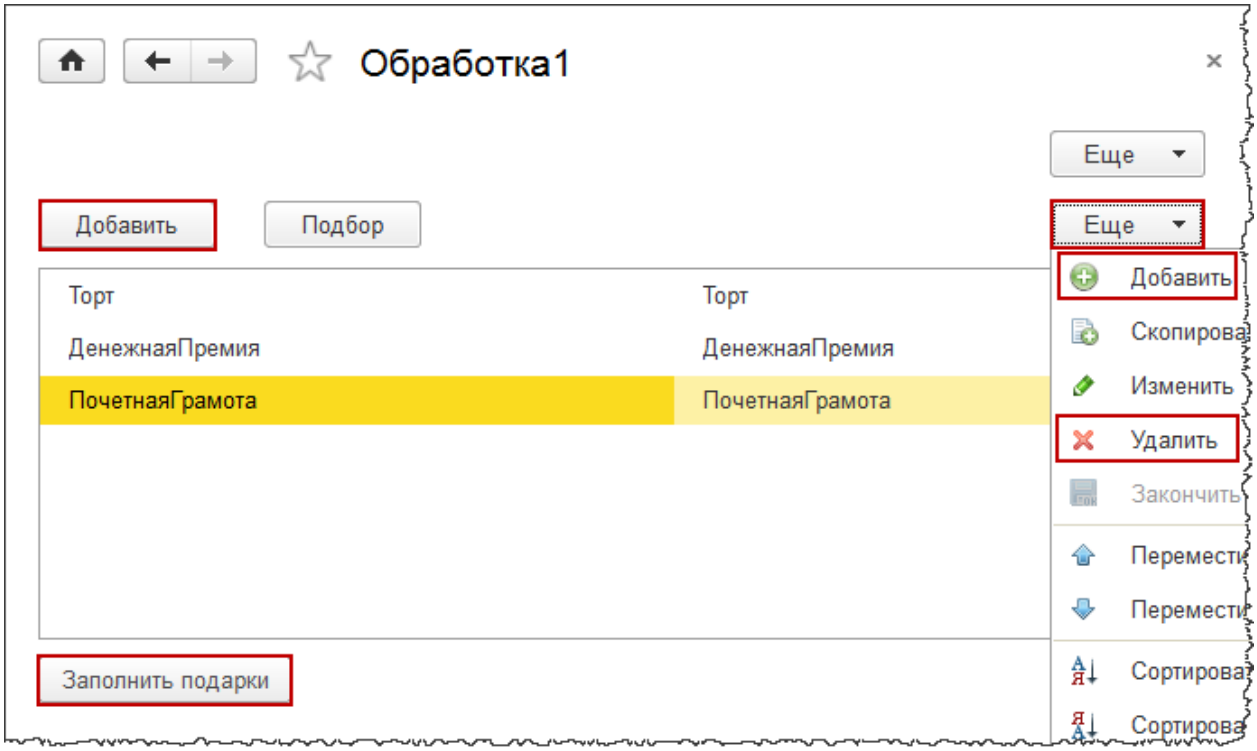
Создаем новую команду *ЗапонитьПодарки*, переносим на форму и определяем для нее обработчик действия.



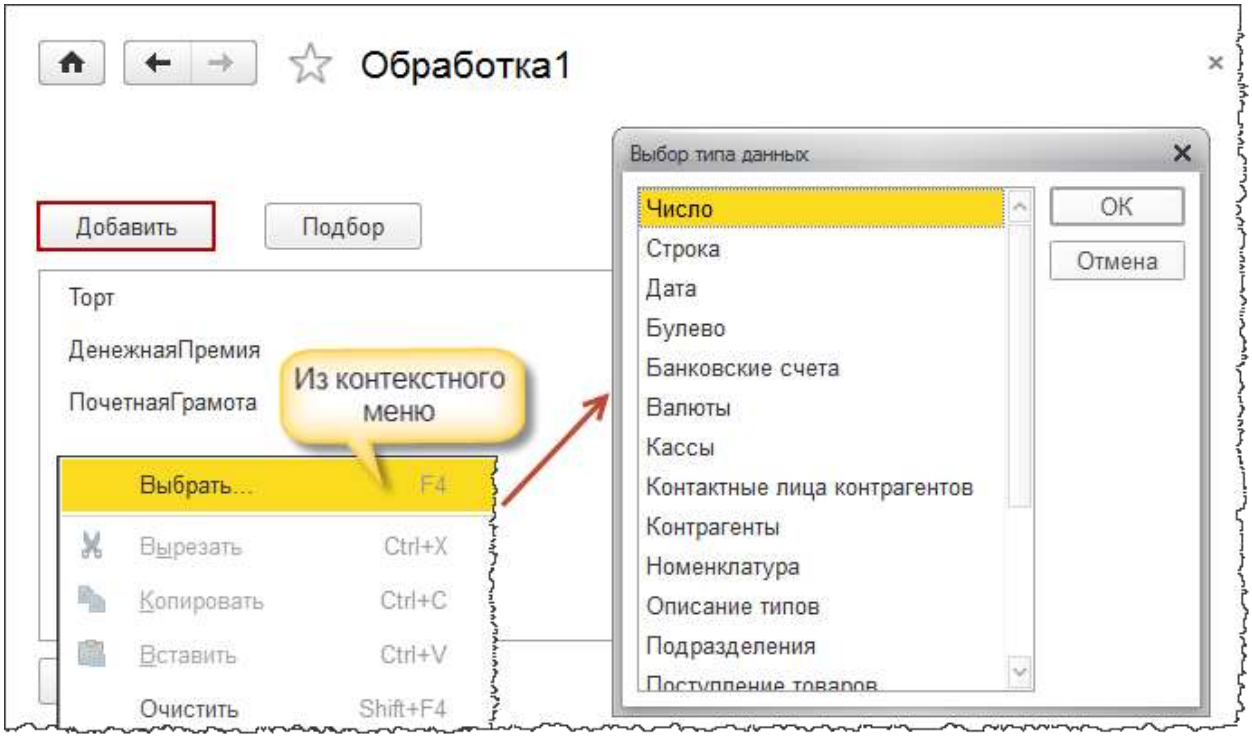
Обработчик действия в модуле формы:

```
&НаКлиенте
Процедура ЗаполнитьПодарки(Команда)

    // Создаем список значений
    Подарки = Новый СписокЗначений;
    Подарки.Добавить("Торт");
    Подарки.Добавить("ДенежнаяПремия");
    Подарки.Добавить("ПочетнаяГрамота",);

КонецПроцедуры
```

В пользовательском режиме, при нажатии в форме обработки кнопки *Заполнить подарки,* появится *заполненный* список.



При желании список можно редактировать: какие-то элементы добавить, какие-то – удалить.



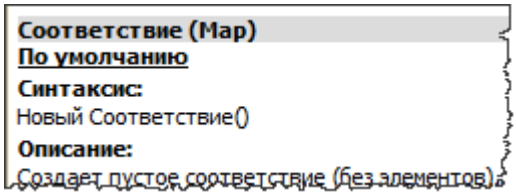
Универсальная коллекция *Соответствие*

Данная коллекция очень схожа со *Структурой*. Также, как и *Структура*, *Соответствие* представляет собой наборы значений, которые состоят их ключа и самого значения.

Главное отличие в том, что в качестве *Ключа* может указываться любой тип данных, равно как и для значения. В виду этой особенности обращаться к значению соответствия необходимо по индексу, в качестве значения индекса указывается значение ключа.

В качестве ключа может быть тип данных, отличающихся от строки. Свойства и методы работы с *Соответствием* практически такие же, как у *Структуры.*

Конструктор *Соответствия*, в отличии от *Структуры*, не содержит возможности указания параметров.



Пример использования:

```
НашеСоответствие = Новый Соответствие;
НашеСоответствие.Вставить(НачалоДня(ТекущаяДата()), "Вторник");
НашеСоответствие.Вставить(НачалоДня(ТекущаяДата())-24*3600, "Понедельник");
Сообщить(НашеСоответствие[НачалоДня(ТекущаяДата())]);
```

Соответствие удобно применять тогда, когда необходимо связать какие-либо две структуры. Например, каждой строке табличной части необходимо сопоставить строку из таблицы значений.

В этом случае в качестве ключа *Соответствия* используется строка табличной части и указывается соответствующее значение.

При вставке элементов в коллекцию *Соответствие* помимо метода *Вставить(<Ключ>,<Значение>)* существует другой способ вставки значения – это использование обычного оператора присваивания.

Например: *НашеСоответствие = Новый Соответствие;*

Соответствие[777] = 999;

Т.е. если элемент в коллекции не присутствовал, то с помощью оператора присваивания он будет добавлен, а если присутствовал, то будет обновлен. Это является отличием от *Структуры*.

Универсальная коллекция Таблица значений

ТаблицаЗначений представляет из себя таблицу с произвольным количеством строк и произвольным количеством колонок. На пересечении могут храниться значения любого типа данных. При необходимости колонки можно типизировать, т. е. определить в какой колонке какой тип данных хранится.

Можно оставить колонки нетипизированными, тогда в одной колонке в разных строках могут храниться значения разных типов.

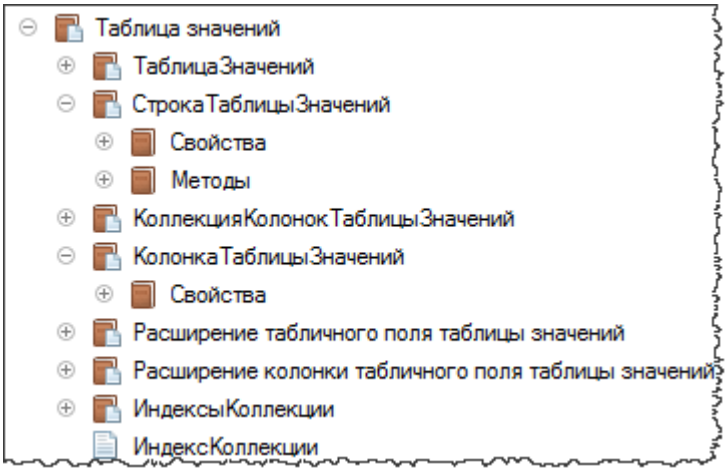
Отличия *ТаблицыЗначений* от двухмерного *Массив*а:

- это объект, с которым может работать пользователь (таблицу значений можно вывести на экран, пользователь может ее заполнять, в дальнейшем введенные данные можно читать);
- построение индексов для быстрого поиска;
- клонирование, заполнение определенным значением всей колонки, выгрузка все колонки в массив.

ТаблицаЗначений используется как некий буфер хранения информации. *ТаблицаЗначений* возвращается и принимается как параметр многими методами системы. К *Таблице значений* возможно построить запрос.

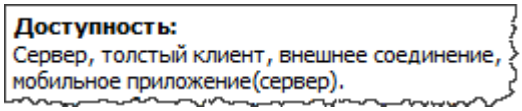
Итак, *ТаблицаЗначений* состоит из набора строк и набора колонок. И строки, и колонки представляют собой коллекции.

Т.е. внутри коллекции *ТаблицаЗначений* есть еще две коллекции. Обратимся к синтакс-помощнику и найдем *ТаблицуЗначений*.



Поддерживаемые типы данных: сама *ТаблицаЗначений,* которая состоит из строк. Каждая строка представлена типом данных *СтрокаТаблицыЗначений*, у которой есть свои свойства и свои методы. Имеется *КоллекцияКолонок ТаблицыЗначений*. *КолонкаТаблицыЗначений* также обладает определенными свойствами.

Важный момент! Процедура, которая формирует ТаблицуЗначений, должна компилироваться *& НаСервере*.



Прежде, чем начать работать с *ТаблицейЗначений*, необходимо определить, какие в ней будут содержаться колонки (т.е. создать их). Синтаксис:

Добавить(<Имя>, <Тип>, <Заголовок>, <Ширина>)

<Имя> (необязательный)

Тип: Строка.

<Тип> (необязательный)

Тип: ОписаниеТипов

<Заголовок> (необязательный)

Тип: Строка.

<Ширина> (необязательный)

Тип: Число.

Например:

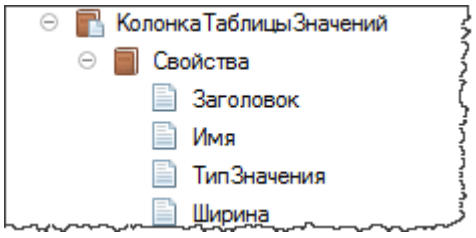
```
&НаСервере
Процедура СозданиеТаблицыЗначений()

ТЗ = Новый ТаблицаЗначений;
ТЗ.Колонки.Добавить("ДеньНедели", Новый ОписаниеТипов("Строка") );
ТЗ.Колонки.Добавить("Сумма");

Строка = ТЗ.Добавить();
Строка.ДеньНедели = "Понедельник";
Строка.Сумма = 2000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка["Сумма"] = 3000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка.Сумма = 1000;

КонецПроцедуры
```

Для вызова данной процедуры будем использовать команду.

```
&НаКлиенте
Процедура КомандаВызоваСозданияТЗ(Команда)
    СозданиеТаблицыЗначений();
КонецПроцедуры
```

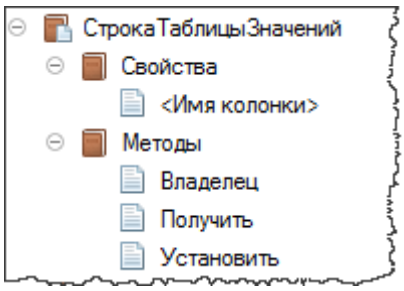
В описании ТаблицыЗначений в качестве элементов коллекции выступают именно *СтрокиТаблицыЗначений*.

В отличии от колонок, которые состоят только из свойств (*Имя*, *Тип*, *Заголовок*, *Ширина*), в *СтрокеТаблицыЗначений* существуют как свойства (обращение по имени колонки), так и методы (можно получать и устанавливать значение, работать с владельцами).



Чтобы добавить новую строку в таблицу нужно использовать метод либо *Добавить()*, либо *Вставить(<Индекс>)*. Во втором случае следует указать, на какую позицию нужно выставлять требуемую строку.

Чтобы присвоить значение колонке, мы через точку обращаемся по имени колонки или по индексу (с помощью квадратных скобок).



Для заполнения ТаблицыЗначений могут использоваться следующие методы:

Очистить() – для удаления всех строк из ТаблицыЗначений.

ЗаполнитьЗначения(<Значение>, <Колонки>) – позволяет заполнить все колонки, либо выбранные колонки одним значением.

ЗагрузитьКолонку(<Массив>, <Колонка>) – загружает колонку из массива.

ВыгрузитьКолонку(<Колонка>) – выгружает колонку в массив.

Два последних метода удобно использовать, когда нужно перебросить колонку из одной таблицы значений в другую.

Скопировать(<Строки>, <Колонки>) – позволяет на основании существующей таблицы создать новую *ТаблицуЗначений*, при этом указывать не все строки и все колонки, а только некоторые из них. Возвращаемое значение – *ТаблицаЗначений*.

Можно скопировать структуру *ТаблицыЗначений*. Для этого существует соответствующий метод *СкопироватьКолонки(<Колонки>)*. Мы получим пустую *ТаблицуЗначений* с требуемой структурой.

В *ТаблицеЗначений* существует метод *Итог(<Колонка>)*. Можно указать ту колонку, в которой нужно просуммировать числовые величины. Применительно к ранее показанному коду в Табло можно рассчитать значение: *ТЗ.Итог(“Сумма”)*.

```
&НаСервере
Процедура СозданиеТаблицыЗначений()

ТЗ = Новый ТаблицаЗначений;
ТЗ.Колонки.Добавить("ДеньНедели", Новый ОписаниеТипов("Строка") );
ТЗ.Колонки.Добавить("Сумма");

Строка = ТЗ.Добавить();
Строка.ДеньНедели = "Понедельник";
Строка.Сумма = 2000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка["Сумма"] = 3000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка.Сумма = 1000;

КонецПроцедуры
```

Табло

Выражение	Значение	Тип
ТЗ.Итог("Сумма")	6 000	Число

В ТаблицеЗначений существует возможность сгруппировать (свернуть) числовые величины по одинаковым значениям определенных колонок с помощью метода *Свернуть(<КолонкиГруппировок>, <КолонкиСуммирования>)* .

Применительно к ранее показанному коду в Табло можно рассчитать значение: *ТЗ.Свернуть(“ДеньНедели”,“Сумма”).*

```
&НаСервере
Процедура СозданиеТаблицыЗначений()

ТЗ = Новый ТаблицаЗначений;
ТЗ.Колонки.Добавить("ДеньНедели", Новый ОписаниеТипов("Строка") );
ТЗ.Колонки.Добавить("Сумма");

Строка = ТЗ.Добавить();
Строка.ДеньНедели = "Понедельник";
Строка.Сумма = 2000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка["Сумма"] = 3000;

Строка = ТЗ.Добавить();
Строка["ДеньНедели"] = "Вторник";
Строка.Сумма = 1000;
ТЗ.Свернуть("ДеньНедели","Сумма");

КонецПроцедуры
```

ТаблицаЗначений

Количество элементов: 2

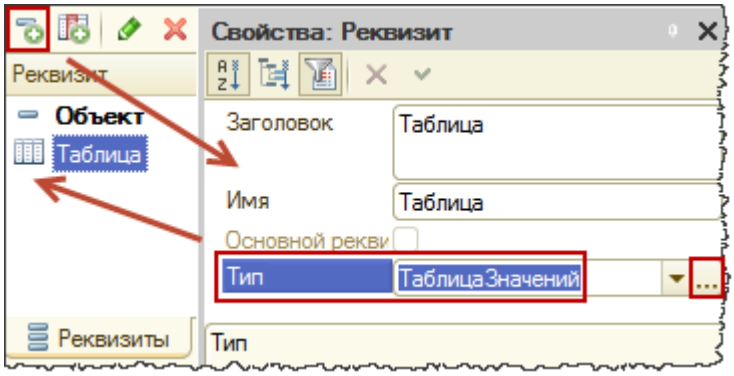
Закрыть

Справка

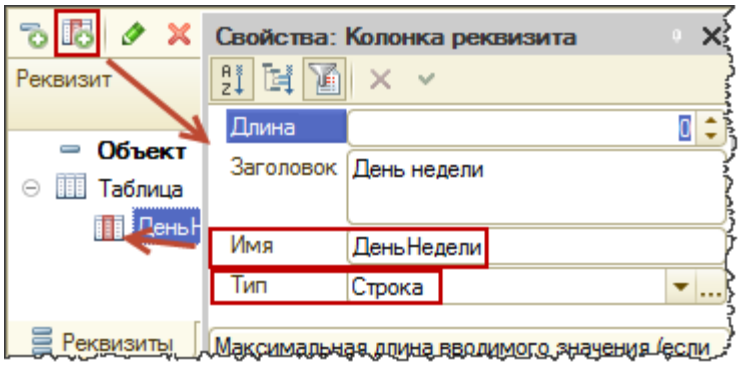
Индекс	Значение элемента	Тип элемента	ДеньНедели	Сумма
0	СтрокаТаблицыЗн...	СтрокаТаблицыЗн...	"Понедельник"	2 000
1	СтрокаТаблицыЗн...	СтрокаТаблицыЗн...	"Вторник"	4 000

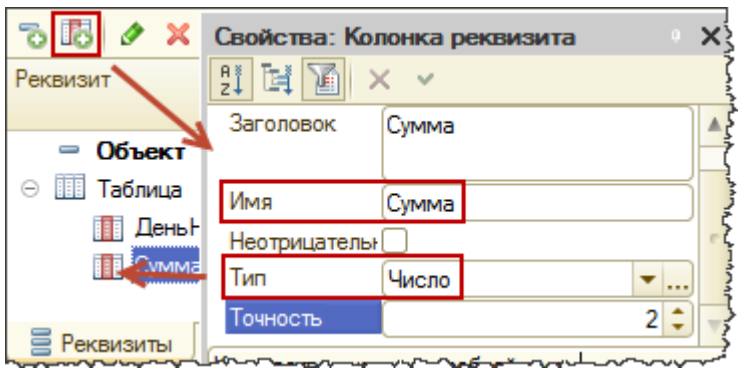
ТаблицуЗначений можно показать на пользовательском экране, чтобы с ней можно было совершать какие-либо действия. Но в отличии от *СпискаЗначений* из программного кода нельзя просто так вызвать таблицу на экран.

Чтобы отобразить *ТаблицуЗначений* на экране, создадим реквизит формы и присвоим ему тип данных *ТаблицаЗначений*.

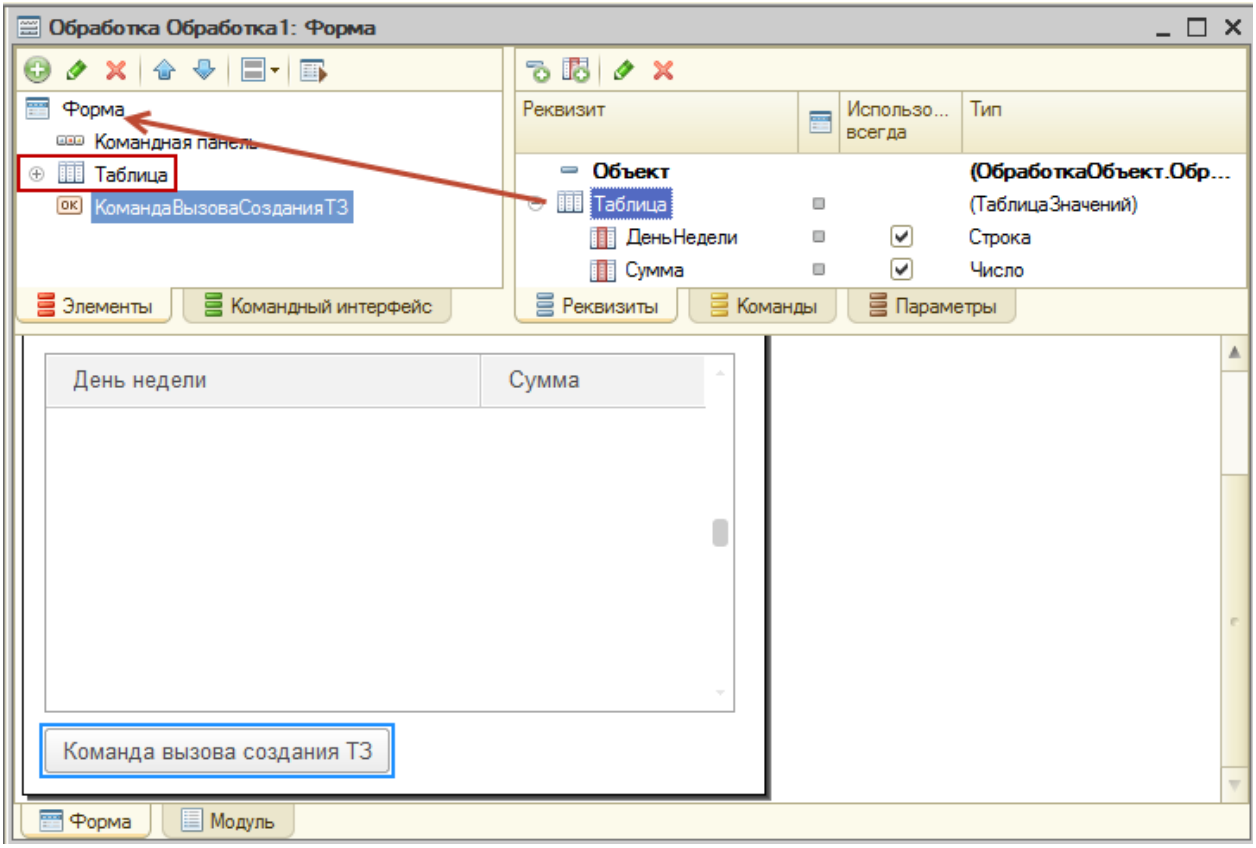


Далее у этого объекта («Таблица») нужно будет создать колонки с соответствующим именем и типом данных.





После чего полученную таблицу следует вывести на форму.



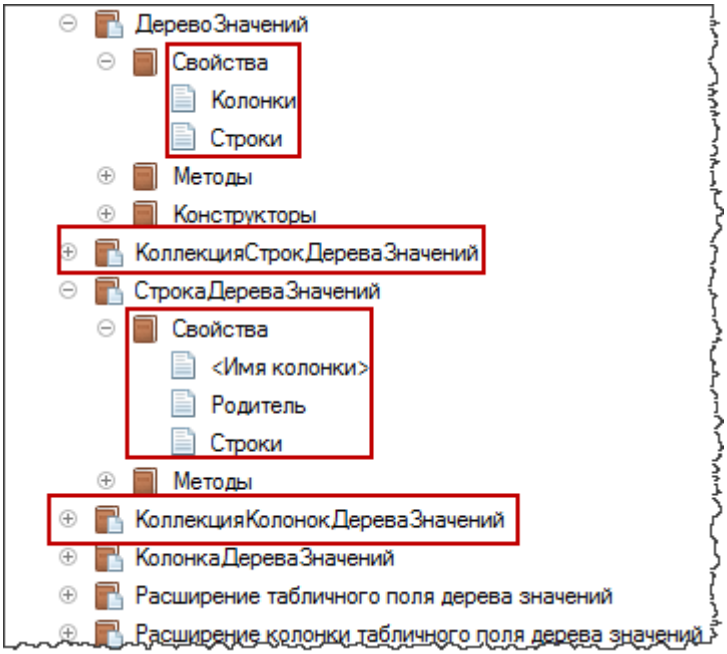
В модуле формы в конце ранее составленного алгоритма (в Процедуре *СозданиеТаблицыЗначений*) следует дописать:

ЗначениеВДанныеФормы(ТЗ, Таблица);

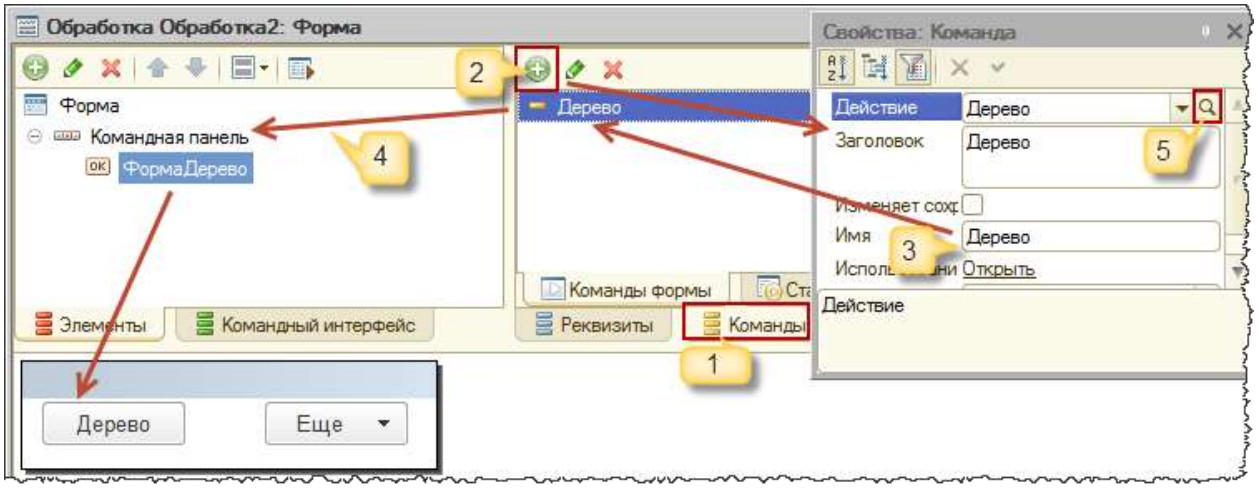
Универсальная коллекция Дерево значений

ДеревоЗначений универсальная коллекция, которая очень похожа на *ТаблицуЗначений*. Отличие от таблицы заключается в том, что строки дерева могут быть подчинены друг другу, т.е. может быть образована некая иерархия.

ДеревоЗначений тоже может быть отражено на экране. Дерево значений в явном виде состоит из коллекции строк и коллекции колонок. В дереве существуют такие два свойства как *Строки* и *Колонки*. Поскольку строки могут быть подчинены друг другу, то для каждой строки может быть указан *Родитель*, а также подчиненные ей строки.



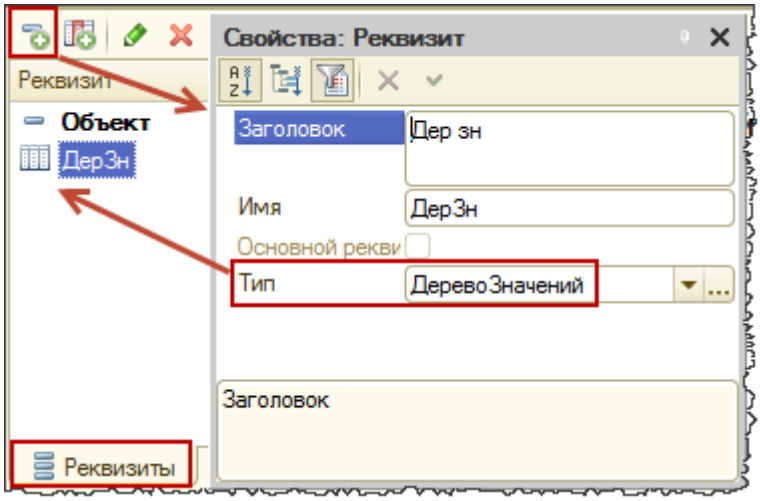
Создадим соответствующую команду *Дерево* и ее процедуру обработки.



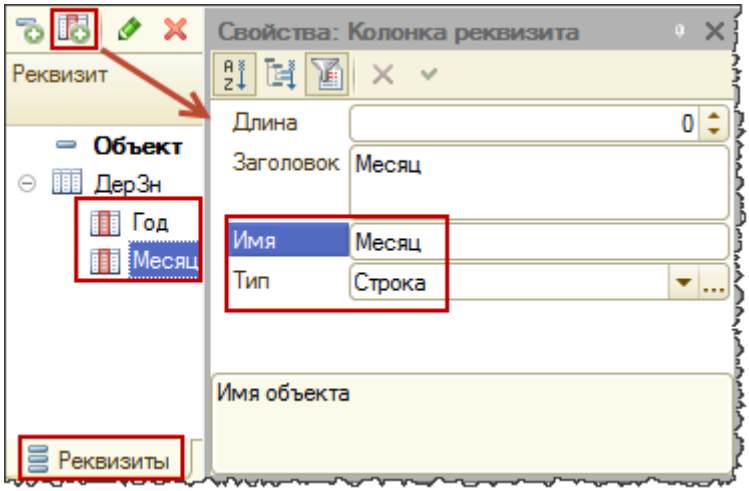
Создадим *ДеревоЗначений* в котором одна родительская строка и две подчиненные.

```
&НаКлиенте
Процедура Дерево(Команда)
    ДеревоНаСервере();
КонецПроцедуры

&НаСервере
Процедура ДеревоНаСервере()

ДеревоЗн = Новый ДеревоЗначений;

ДеревоЗн.Колонки.Добавить("Год");
ДеревоЗн.Колонки.Добавить("Месяц");
ДеревоЗн.Колонки.Добавить("Факт");

Строка = ДеревоЗн.Строки.Добавить();
Строка.Год = 2014;
Строка.Месяц = "Все месяцы";

//Добавление подчиненных строк

ПодчиненнаяСтрока = Строка.Строки.Добавить();
ПодчиненнаяСтрока.Месяц = "Январь";
ПодчиненнаяСтрока = Строка.Строки.Добавить();
ПодчиненнаяСтрока.Месяц = "Февраль";

КонецПроцедуры
```

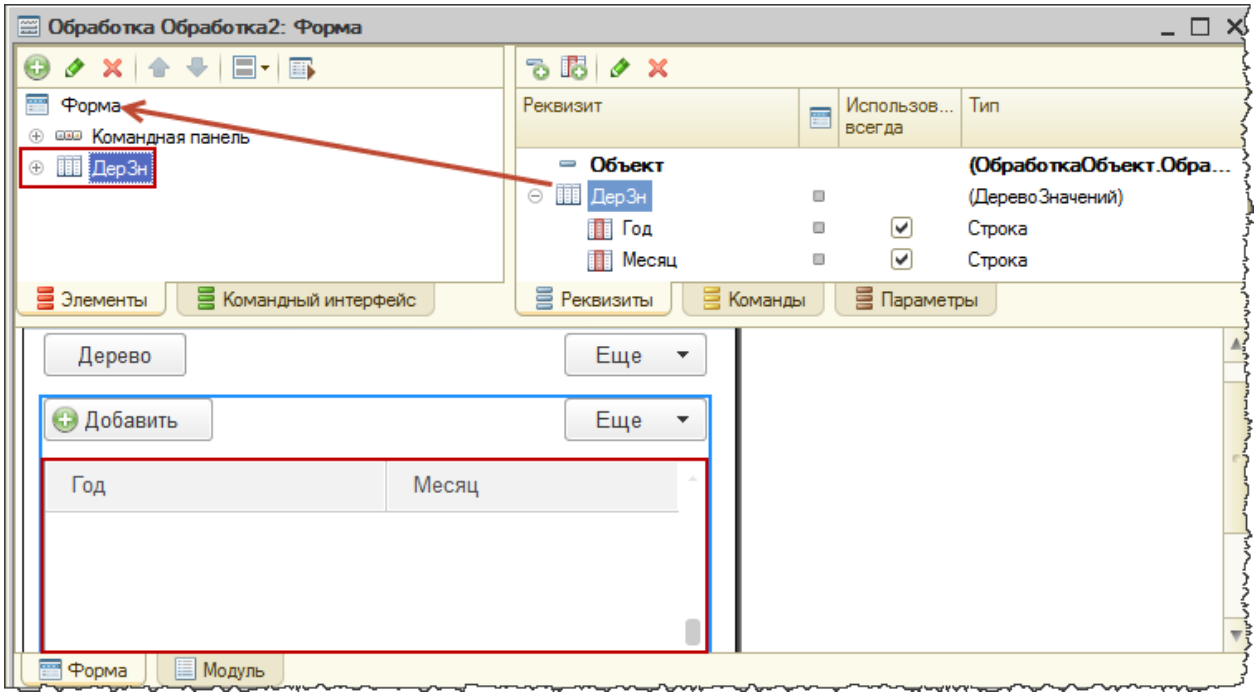
Создадим реквизит формы *ДерЗн* (тип данных – *ДеревоЗначений*).



Для этого реквизита создадим колонки *Год* и *Месяц*.



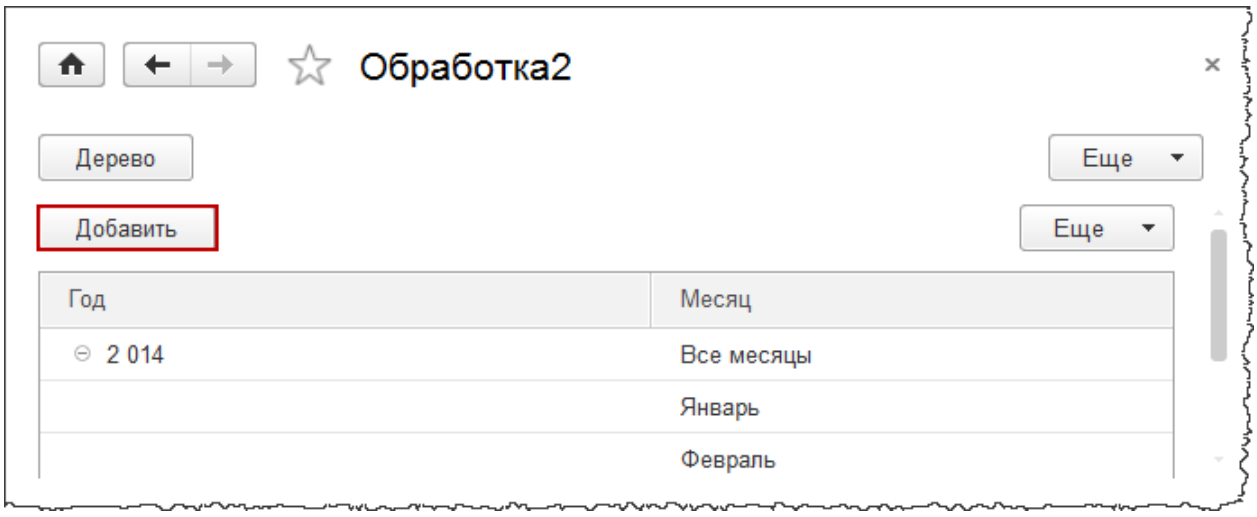
Переместим соответствующий элемент *ДерЗн* на форму.



В конце Процедуры *ДеревоНаСервере()* допишем:

ЗначениеВДанныеФормы(ДеревоЗн, ДерЗн);

Проверим, что получилось в пользовательском режиме.



С помощью кнопки *Добавить* можно добавлять новые строки. Они могут также образовывать иерархию.

Чтобы обойти все элементы дерева значений, нам понадобится использовать рекурсию, т.е. вызов процедуры самой из себя. Например, обработка дерева значений может выглядеть так:

```
Строки = ДеревоЗн.Строки;

КонецПроцедуры

&НаСервере
Процедура ОбработатьДерево (Строки)

Для каждого Строка из Строки Цикл
Сообщить (Строка(Строка.Год)+ " " +Строка.Месяц);
ОбработатьДерево(Строка.Строки);
КонецЦикла;

КонецПроцедуры
```

Артемов Артем,

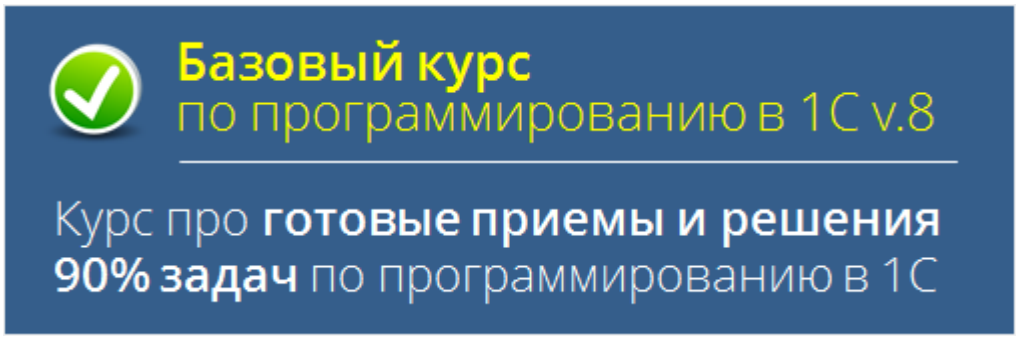
г. Москва

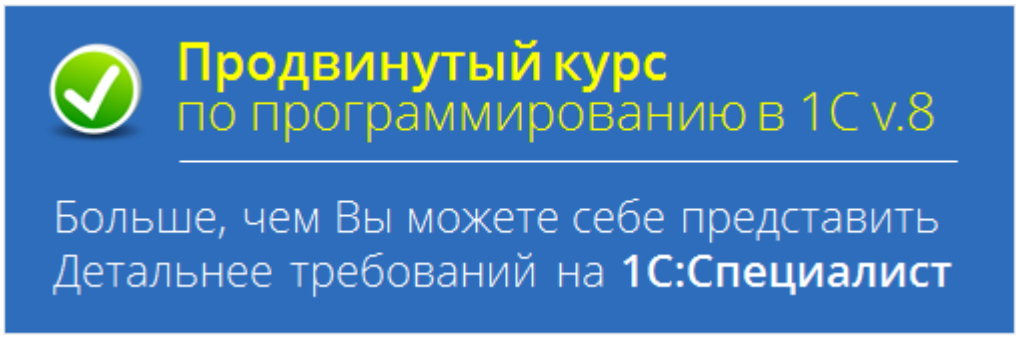
Дополнительные материалы

Все статьи проекта **Курсы-по-1С.рф**: **http://курсы-по-1с.рф/blog/articles/**

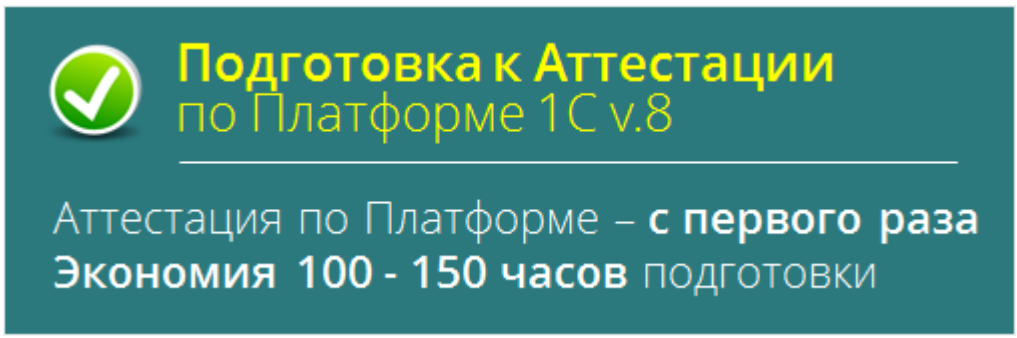
Курсы по программированию в 1С v.8

Базовый и Продвинутый курсы по Программированию на Платформе 1С 8
http://www.Spec8.ru/





«Курс по подготовке к Аттестации по Платформе 1С 8.2 / 8.3»
http://курсы-по-1с.рф/dev-attestation/



Бесплатные материалы проекта:
http://курсы-по-1с.рф/free/

Электронная почта: support@Kursy-po-1C.ru
Телефоны: **8-800 1000-613**, +7(495) 7777-096