**Лабораторная работа №17**

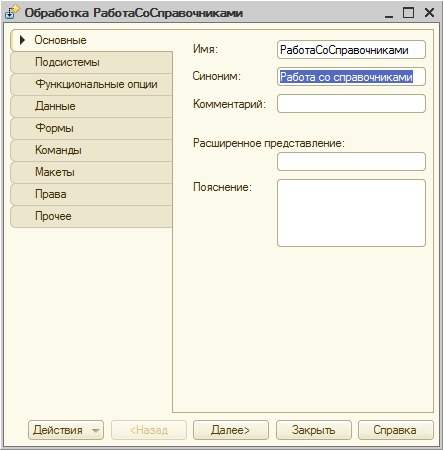
**Тема:** Программная работа со справочниками, обработки, простые отчеты.

**Цель работы:** Исследовать механизмы программной работы со справочниками и создания обработок и простых отчетов

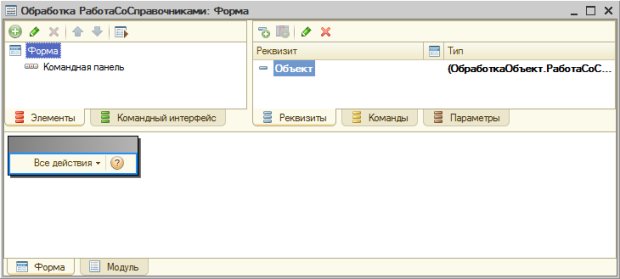
**Последовательность выполнения работы:**

### Обработка, программная работа со справочниками

Начнем с создания обработки, которая выводит имена всех справочников, имеющихся в системе. Для этого добавим новую обработку в ветви **Обработки** *дерева конфигурации*. Назовем ее **РаботаСоСправочниками**, [рис. 5.1](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=1#image.5.1).

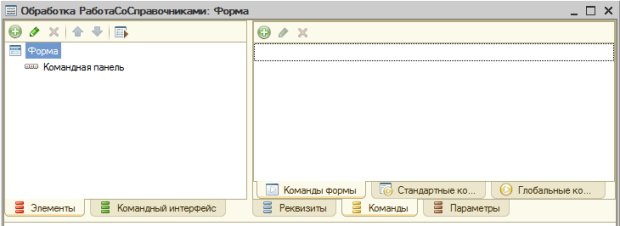
  
**Рис. 5.1.**Создание обработки

Включим новую обработку в состав подсистемы **Администрирование** на закладке **Подсистемы**. Перейдем на закладку **Формы** и создадим форму обработки. Наша обработка не имеет реквизитов – сразу после запуска конструктора формы обработки, мы можем нажать на кнопку **Готово** и увидим пустую форму обработки, [рис. 5.2](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=1#image.5.2).

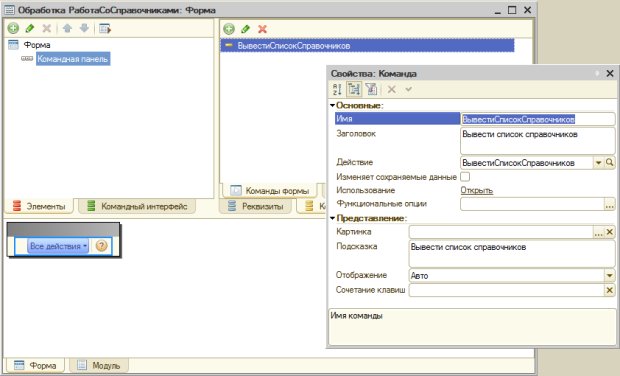
[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_02.jpg)  
**Рис. 5.2.**Форма обработки

Задачу вывода данных *по* результатам работы обработки мы переложим на плечи системы – будем выводить их с помощью стандартного механизма сообщений, которым мы уже не раз пользовались. А вот команды, которые будут выполняться обработкой, нам нужно будет создать самостоятельно.

Перейдем на вкладку **Команды** в окне редактора форм. После этого нам будут доступны еще несколько вкладок, нас интересует первая из них – **Команды формы**, [рис. 5.3](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=1#image.5.3).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_03.jpg)  
**Рис. 5.3.**Переход к командам формы обработки

*Список* команд формы пуст – нам нужно создать собственную команду. Нажмем на кнопку **Добавить** в верхней части панели **Команды формы**, назовем ее **ВывестиСписокСправочников**, в окне свойств команды нажмем на кнопку с увеличительным стеклом в *поле* свойства **Действие** – в модуле формы будет создана процедура для этой команды, [рис. 5.4](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=1#image.5.4).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_04.jpg)  
**Рис. 5.4.**Настройка команды

В *модуль* формы был добавлен такой код:

&НаКлиенте

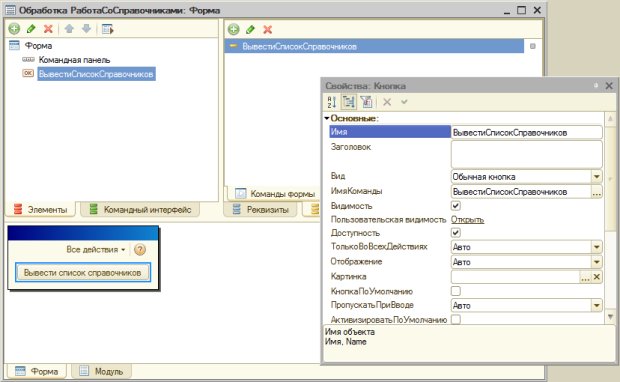
Процедура ВывестиСписокСправочников(Команда)

// Вставить содержимое обработчика.

КонецПроцедуры

То, что мы добавили в обработку команду, еще не означает автоматическое добавление на форму команды, например, кнопки, нажатие которой приведет к выполнению команды. Добавить такую кнопку на форму можно несколькими способами. Во-первых, мы можем просто перетащить команду из панели **Команды формы** на панель **Элементы** – на форме появится кнопка **Вывести список справочников**, а напротив команды – серый квадратик, говорящий о присутствии элемента управления, связанного с командой, на форме.

Во-вторых, в *список* элементов формы можно добавить кнопку (кнопка **Добавить** в командной панели закладки **Элементы**) и задать свойства кнопки, в частности, в свойстве **ИмяКоманды** выбрать нужную команду. После добавления кнопки и настройки ее связи с командой, редактор форм приобрел вид, показанный на [рис. 5.5](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=1#image.5.5).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_05.jpg)  
**Рис. 5.5.**Настройка кнопки

Теперь приступим к редактированию кода. Код команды выполняется на клиенте, нам же нужно работать с базой данных, то есть – объявить серверную процедуру или функцию. В итоге у нас получился следующий код:

&НаКлиенте

Процедура ВывестиСписокСправочников(Команда)

ВывестиИменаСправочников();

КонецПроцедуры

Процедура ВывестиИменаСправочников()

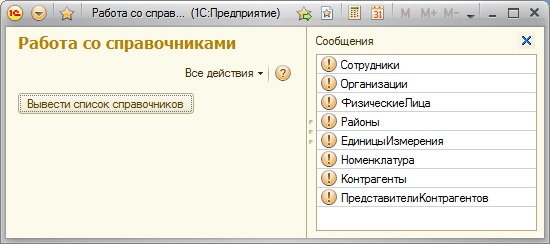
Для каждого Справочник из Метаданные.Справочники Цикл

Сообщить (Справочник.Имя);

КонецЦикла;

КонецПроцедуры

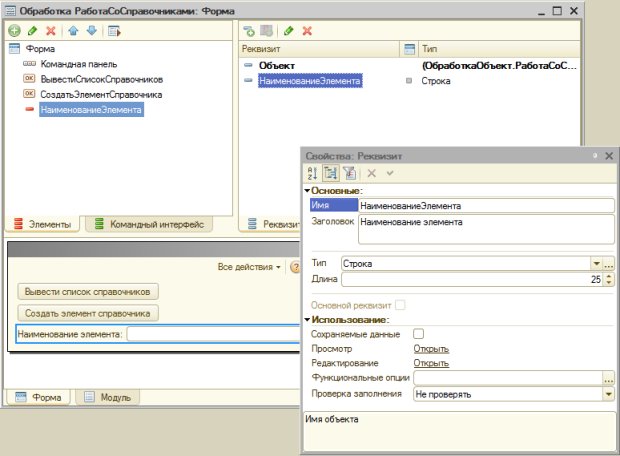
Обратите внимание на то, что объявляя процедуру **ВывестиИменаСправочников()**, мы не указываем директиву компиляции – *по* умолчанию подставляется *директива* **&НаСервере**. В процедуре мы перебираем коллекцию **Метаданные.Справочники**. Коллекция **Метаданные** относится к глобальному контексту и дает *доступ* к структуре метаданных конфигурации. Эта коллекция имеет тип **ОбъектМетаданныхКонфигурация**. С помощью коллекции **Метаданные** мы получаем *доступ* к коллекции **Справочники**. Эта коллекция, в свою *очередь*, имеет тип **КоллекцияОбъектовМетаданных** – в нее входят объекты, которые описывают все справочники, входящие в систему. При обходе коллекции мы получаем ОбъектМетаданных: Справочник, посредством которого можем обращаться к метаданным этого объекта. В частности, мы получаем имена справочников и выводим их в окно сообщений, [рис. 5.6](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=2#image.5.6).



**Рис. 5.6.**Вывод списка справочников

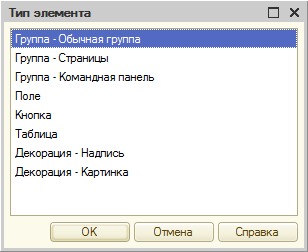
Рассмотрим еще одну задачу. Нужно программно создать элемент справочника с заданными параметрами. На верхнем уровне типов данных, которые имеют *отношение* к справочникам, находится *объект* **Справочники**, имеющий тип **СправочникиМенеджер**. С его помощью можно обращаться к отдельным справочникам, через их объекты **СправочникМенеджер**. При работе с объектом типа **СправочникиМенеджер** используется свойство глобального контекста **Справочники**.

Обращение к объектам **СправочникМенеджер** возможно *по* имени справочника, заданному в конфигурации. Мы собираемся программно создать элемент с наименованием, которое задаст *пользователь* в форме обработки. Для этого добавим в *список* команд формы новую – назовем ее **СоздатьЭлементСправочника**, создадим ее процедуру, добавим ее на форму. Добавим новый *реквизит* в *список* реквизитов, назовем его **НаименованиеЭлемента**, зададим тип – **Строка**, *длина* **25**, так же переместим *реквизит* в область **Элементы** – там он будет представлен в виде текстового поля, [рис. 5.7](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=2#image.5.7).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_07.jpg)

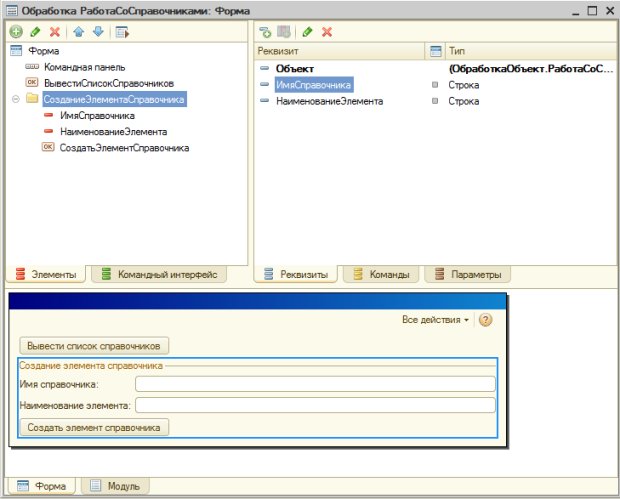
**Рис. 5.7.**Настройка нового реквизита формы

Добавим еще один *реквизит* – назовем его **ИмяСправочника**, тип **Строка**, *длина* – **100**. Сюда *пользователь* будет вводить имя справочника, в котором он хочет создать новый элемент. На нашей форме теперь имеются три логически связанных элемента. Удобно объединить их в одну группу, чтобы *пользователь* сразу мог понять, что они работают вместе. Для этого можно сгруппировать элементы. В командной панели вкладки **Элементы** нажмем на кнопку **Добавить**, появится окно – **Тип элемента** ([рис. 5.8](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=2#image.5.8).), среди списка элементов, представленных в котором, можно найти несколько видов групп.



**Рис. 5.8.**Добавление новой группы на форму

Обычная *группа* позволяет визуально разделить элементы, находящиеся на форме. Добавим на форму новую группу, назовем ее **СозданиеЭлементаСправочника**, перетащим в нее *элементы управления*, относящиеся к этой группе. Результат реорганизации элементов показан на [рис. 5.9](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=2#image.5.9).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_09.jpg)

**Рис. 5.9.**Добавление новой группы на форму

Теперь займемся кодом. Нам, в *дополнение* к клиентской процедуре команды **СоздатьЭлементСправочника**, понадобится серверная процедура или *функция*, которая и занимается созданием элемента. Обратиться к объекту **СправочникМенеджер** для конкретного справочника можно различными способами. Предположим, мы заранее знаем, с каким справочником нам нужно работать (например, это – справочник **Номенклатура**). Для того, чтобы вызвать метод этого справочника **СоздатьЭлемент**, нам понадобится такая конструкция:

НовыйЭлемент=Справочники.Номенклатура.СоздатьЭлемент();

В данном случае происходит следующее. Посредством объекта **СправочникиМенеджер** (**Справочники**) мы получаем *доступ* к объекту **СправочникМенеджер** для справочника **Номенклатура** и выполняем его метод **СоздатьЭлемент**. Этот метод возвращает нам *объект* типа **СправочникОбъект** (*доступ* к нему возможен через переменную **НовыйЭлемент**).

В нашем случае имя справочника задает *пользователь*, оно нам заранее неизвестно. В том случае, если имя справочника для вышеописанной конструкции будет, перед обращением к менеджеру справочника, записано в некую переменную, мы можем использовать такую конструкцию (в нашем случае имя справочника хранится в текстовой переменной **ИмяСправочника**):

НовыйЭлемент = Справочники[ИмяСправочника].СоздатьЭлемент();

**Оператор[…]**, который используется в данной конструкции, заменяет конструкцию с точкой и жестко заданным именем справочника.

После того, как мы получили переменную типа **СправочникОбъект**, мы можем настроить необходимые свойства конкретного элемента справочника (в нашем случае – наименование) и записать элемент. Вот, как выглядит результирующий код:

&НаКлиенте

Процедура СоздатьЭлементСправочника(Команда)

КодНовогоЭлемента=СоздатьЭлементСправочникаНаСервере();

Сообщить("В справочнике "+ИмяСправочника+" создан элемент

"+НаименованиеЭлемента + " с автоматически присвоенным кодом:

"+КодНовогоЭлемента);

КонецПроцедуры

Функция СоздатьЭлементСправочникаНаСервере()

НовыйЭлемент = Справочники[ИмяСправочника].СоздатьЭлемент();

НовыйЭлемент.Наименование=НаименованиеЭлемента;

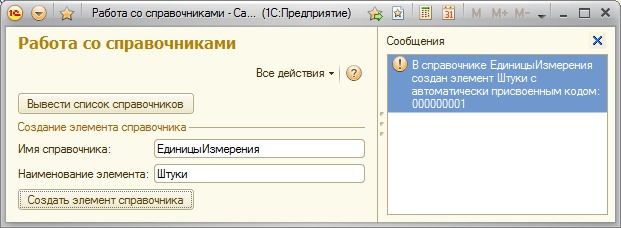
НовыйЭлемент.Записать();

Возврат (НовыйЭлемент.Код);

КонецФункции

Обратите внимание на то, что мы в серверной процедуре обращаемся к реквизитам формы напрямую – они доступны и на сервере и на клиенте, так как *функция*, в которой они вызываются – это *функция*, объявленная с используемой *по* умолчанию директивой **&НаСервере**. Если бы мы в подобной ситуации попытались воспользоваться серверной внеконтекстной функцией (*директива* **&НаСервереБезКонтекста**) – обращаться к контексту формы (к ее реквизитам), мы не смогли бы. Вместо того, чтобы пользоваться стандартными механизмами обмена данными с сервером (а при вызове серверной процедуры на *сервер* передаются данные от клиента о состоянии формы), нам пришлось бы организовывать передачу этих данных вручную через параметры методов. Серверная внеконтекстная *функция* позволила бы снизить объем данных, передаваемых с клиента на *сервер* и обратно. Но она, в то же время, способна выполнять те же действия с базой, что и *функция*, объявленная с ключевым словом **&НаСервере**.

Вот, каковы результаты работы этого кода, [рис. 5.10](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=3#image.5.10).



**Рис. 5.10.**Создание нового элемента справочника

*Функция* **СоздатьЭлементСправочникаНаСервере** создает новый элемент, заполняет его свойство **Наименование**, после чего записывает его и возвращает код нового элемента. Код формируется системой автоматически. Если заглянуть в справочник **ЕдиницыИзмерения** – там, действительно, будет создан новый элемент с заданным нами наименованием.

Продолжим наши примеры программной работы со справочниками. Нам нужно реализовать автоматическую пометку всех элементов (но не групп) справочника на удаление. Создадим новую команду **ПометитьНаУдалениеВсеЭлементыСправочника**. После *создания процедуры*, связанной с этой командой и серверной процедуры, выполняющей работу с базой. У нас получился такой код:

&НаКлиенте

Процедура ПометитьНаУдалениеВсеЭлементыСправочника(Команда)

ПометитьНаУдаление();

КонецПроцедуры

Процедура ПометитьНаУдаление()

СчетчикПомеченных = 0;

Выборка = Справочники[ИмяСправочника].Выбрать();

Пока Выборка.Следующий() Цикл

Элемент=Выборка.ПолучитьОбъект();

Если НЕ Элемент.ЭтоГруппа Тогда

Элемент.УстановитьПометкуУдаления(Истина);

СчетчикПомеченных=СчетчикПомеченных+1;

КонецЕсли;

КонецЦикла;

Сообщить("В справочнике "+ИмяСправочника+" помечено на удаление"+СчетчикПомеченных+" элементов");

КонецПроцедуры

В процедуре **ПометитьНаУдаление()**мы сначала присваиваем 0 переменной **СчетчикПомеченых** – с ее помощью мы будем подсчитывать количество помеченных на удаление элементов справочника. В качестве имени справочника мы используем уже знакомый *по* прошлой процедуре *реквизит* **ИмяСправочника**. Конструкция **Справочники[ИмяСправочника]** позволяет нам обратиться к объекту типа **СправочникМенеджер** для заданного справочника. Этот *объект* имеет метод **Выбрать()**. Метод **Выбрать()** позволяет сформировать выборку элементов справочника *по* заданным условиям. Мы, в данном случае, условий не задаем, то есть в выборку попадают все элементы и группы справочника – метод возвращает *значение* типа **СправочникВыборка**. **СправочникВыборка** не содержит элементов справочника, *объект* этого типа можно считать способом доступа к элементам, способом их перебора. При обращении к выборке обход элементов осуществляется динамически, данные считываются из базы порциями, что позволяет эффективно использовать данный механизм даже для работы с большими справочниками, так как все элементы, входящие в выборку (соответствующие условиям выборки) в *память* не загружаются.

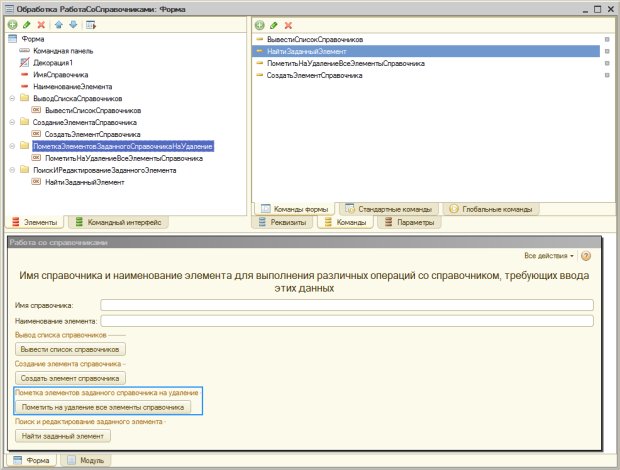
*Команда* **Выборка.Следующий()**, во-первых, возвращает *значение* **Истина** (в нашем случае это приводит к запуску следующей итерации *цикла*), если в выборке выбран следующий элемент, во-вторых, получает следующий элемент выборки. Обращение к этому элементу осуществляется через ту же переменную **Выборка** типа **СправочникВыборка**. Для получения объекта элемента справочника мы пользуемся методом **Выборка.ПолучитьОбъект()** – он возвращает *объект* типа **СправочникОбъект**, с которым мы можем дальше работать. А именно, мы проверяем, не является ли найденный элемент группой, если не является – используем метод **УстановитьПометкуУдаления** объекта типа **СправочникОбъект**. Этот метод принимает один обязательный *параметр*, которые следует первым в списке параметров, а именно – для установки пометки удаления он должен быть установлен в *значение* **Истина** (как в нашем случае), для снятия – в *значение* **Ложь**.

После установки пометки удаления мы увеличиваем *счетчик* **СчетчикПомеченных** и переходим к следующей итерации *цикла*. Когда цикл перебора элементов выборки завершается, мы выводим сообщение о количестве элементов, помеченных на удаление в справочнике, имя которого задано в реквизите **ИмяСправочника**.

Среди объектов, с которыми вы имеете дело, работая со справочниками, вам встретится *объект* типа **СправочникСсылка**. Обычно мы задаем подобный тип (**СправочникСсылка.Контрагенты** и т.д.) при настройке реквизитов других объектов, которые должны хранить некий элемент нужного справочника. На самом деле, элемент хранится в таблице справочника, в базе данных, а *реквизит*хранит лишь ссылку. При программной работе со справочниками мы получаем *объект* **СправочникСсылка**, когда, например, ищем какой-то элемент справочника. Ссылку можно использовать для идентификации элемента, а так же – для перехода к объекту типа **СправочникОбъект**, если тот элемент, на который у нас есть *ссылка*, нужно, например, отредактировать. *Объект* типа **СправочникСсылка** не предназначен для изменения элемента справочника. В свою *очередь*, от **СправочникОбъект** можно перейти к **СправочникСсылка** – у объекта имеется соответствующее *поле* – **Ссылка**.

Рассмотрим пример. В заданном справочнике нужно найти элемент с заданным наименованием (или сообщить, что элемента с таким наименованием в справочнике нет), изменить *регистр* символов в наименовании таким образом, чтобы все буквы были прописными, и сообщить пользователю его код с указанием старого и нового наименования.

Обычным образом добавим в форму обработки новую команду, для указания имени справочника и наименования искомого элемента используем те же реквизиты **ИмяСправочника** и **НаименованиеЭлемента**, реорганизуем *элементы управления* на форме, [рис. 5.11](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=3#image.5.11).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_11.jpg)

**Рис. 5.11.**Переработанная форма

*Поиск*, редактирование заданного элемента и *вывод* необходимых сообщений реализуется с помощью следующего кода:

&НаКлиенте

Процедура НайтиЗаданныйЭлемент(Команда)

НайтиЗаданныйЭлементНаСервере();

КонецПроцедуры

Процедура НайтиЗаданныйЭлементНаСервере()

СсылкаНаЭлемент=Справочники[ИмяСправочника].НайтиПоНаименованию(НаименованиеЭлемента);

Если СсылкаНаЭлемент=Справочники[ИмяСправочника].ПустаяСсылка() Тогда

Сообщить ("В справочнике "+ИмяСправочника+" нет элемента "+НаименованиеЭлемента);

Иначе

Элемент=СсылкаНаЭлемент.ПолучитьОбъект();

СтароеНаименование=Элемент.Наименование;

Элемент.Наименование=ВРег(Элемент.Наименование);

Элемент.Записать();

Сообщить("Элемент справочника "+ИмяСправочника+"

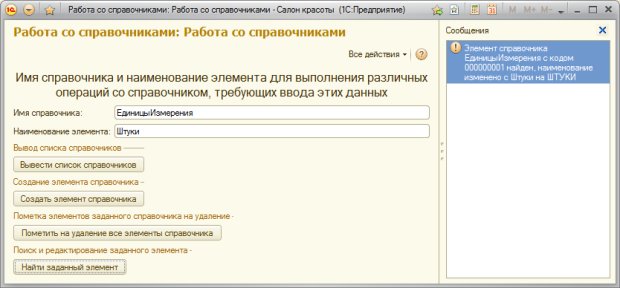
с кодом "+Элемент.Код+" найден, наименование изменено

с "+СтароеНаименование+" на "+Элемент.Наименование);

КонецЕсли;

КонецПроцедуры

В процедуре **НайтиЗаданныйЭлементНаСервер**е() мы обращаемся к методу **НайтиПоНаименованию()** объекта **СправочникМенеджер**, полученному посредством конструкции **Справочники[ИмяСправочника].** Этот метод, среди прочих, принимает обязательный *параметр*, который должен содержать строку с наименованием искомого объекта. Мы передаем ему *реквизит*с искомой строкой. Если метод нашел элемент, наименование которого соответствует этой строке, он вернет ссылку на элемент с типом **СправочникСсылка.ИмяСправочника**. Если нет – будет возвращена пустая *ссылка*. Сравнивая возвращенную ссылку с пустой ссылкой на элемент справочника, который мы обрабатываем, мы принимаем решение о том, сообщить ли пользователю об отсутствии искомого элемента, или, если элемент все же найден, переходим от ссылки на него к объекту (метод **ПолучитьОбъект()** объекта **СправочникСсылка**), выполняем с ним необходимые действия и выводим соответствующее сообщение пользователю. Вот как выглядит работа этого кода, [рис. 5.12](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=3#image.5.12).



**Рис. 5.12.**Результат работы кода по поиску и редактированию элемента справочника

Итак, мы обсудили различные типы данных, которые могут встретиться вам при работе со справочниками. Подведем краткие итоги *по*их основным особенностям и использованию

**СправочникиМенеджер** – доступен через свойство глобального контекста **Справочники**. Предназначен для управления справочниками, позволяет получить *доступ* к объекту **СправочникМенеджер** конкретного справочника.

**СправочникМенеджер** – нужен для управления справочником как объектом конфигурации. С его помощью можно создавать элементы и группы справочника, искать элементы в справочнике, помечать их на удаление, получать выборки элементов справочника.

**СправочникВыборка** – *объект* этого типа предназначен для работы с выборкой элементов справочника, полученной *по* заданным условиям. Для получения выборки используется метод **Выбрать()** объекта **СправочникМенеджер**

**СправочникСсылка** – основная область применения – использование в реквизитах других объектов для указания ссылки на определенный элемент справочника. *Ссылка* – это *идентификатор* элемента. Если имеется *объект* **СправочникСсылка**, а элемент справочника нужно редактировать или выполнять с ним другие подобные действия (*копирование* элемента, например), от ссылки осуществляется переход к объекту типа **СправочникОбъект**.

**СправочникОбъект** – предназначен для манипуляций с отдельным элементом справочника, в частности, для чтения, изменения, добавления, удаления элементов.

Для работы с метаданными справочника можно использовать свойство глобального контекста **Метаданные**, или, например, метод **Метаданные()** объекта типа **СправочникСсылка**. Для работы с метаданными справочника применяется *тип данных***ОбъектМетаданных: Справочник**.

Иногда возникает путаница с понятиями "данные, хранящиеся в справочнике" и "*метаданные* справочника". Данные справочника – это элементы справочника – например, в справочнике **ЕдиницыИзмерения** может храниться элемент с наименованием "Штука", и кодом "0001". *Метаданные* – это, как принято говорить, "данные о данных". То есть, например – это имя справочника, набор его реквизитов, *список* владельцев справочника и так далее. *Метаданные*, другими словами – это то, что мы редактируем, работая в **Конфигураторе**, а данные – это то, с чем мы взаимодействуем, работая со справочником в пользовательском режиме. При работе со справочником как с объектом метаданных, мы можем обращаться к свойствам этого объекта только для чтения – *операции* *по* модификации метаданных производятся в визуальном режиме с помощью **Конфигуратора**. При программной работе с данными справочника, мы имеем полный набор инструментов для управления этими данными.

Есть еще один *тип данных*, имеющий *отношение* к справочникам, о котором мы здесь не упоминали. Это – **СправочникСписок** – он используется для управления списком элементов справочника в табличных полях.

Обсудив программную работу со справочниками, перейдем к разговору об отчетах

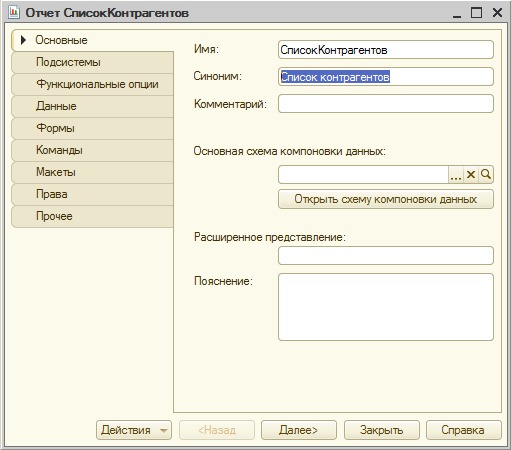
### Простой отчет

Конечная цель любой учетной системы – формирование отчетов. 1С:Предприятие 8.2 предоставляет разработчику множество инструментов для *создания отчетов* – от достаточно простых механизмов, позволяющих создавать несложные отчеты, до комплексных средств, таких, как *система компоновки данных*. Сейчас мы рассмотрим пример создания простого отчета. Нам нужно вывести отчет в виде списка контрагентов *по* группам с указанием наименования контрагента, основного контактного лица и телефона этого контактного лица.

Отчет нужного нам вида можно сформировать различными способами. Так, вполне можно реализовать эту функциональность непосредственно в справочнике **Контрагенты**, добавив в него соответствующие программные *механизмы*, выведя кнопку "**Сформировать список контрагентов**" в форму списка справочника. Можно сделать это с помощью специализированного прикладного объекта **Отчет**. Обычно для создания подобного рода отчетов так и поступают.

Отчет, в нашем случае, будет строиться на основе макета, с областями которого работают в программном коде, формируя готовый отчет. Как правило, если речь идет о неких общих отчетах, их создают в виде отдельных объектов, если же, например, нам нужно создать печатную форму для отдельного элемента справочника или отдельного документа – вполне можно "поместить" всю функциональность требуемого отчета внутри прикладного объекта. В частности, прикладные объекты могут иметь в числе подчиненных объектов макеты, которые и используются при создании отчетов. В то же время, внешний отчет вполне можно использовать и для создания *печатной формы* отдельного элемента справочника или документа.

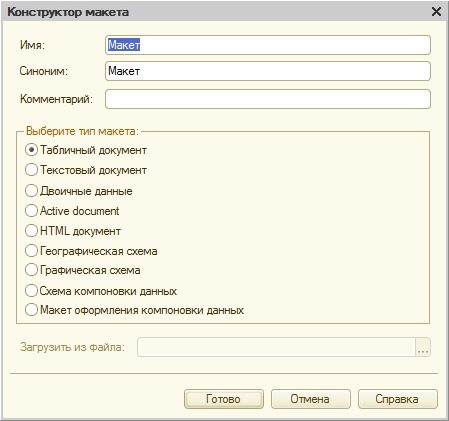
Создадим в ветви *дерева конфигурации* **Отчеты** новый отчет, дадим ему имя **СписокКонтрагентов**, [рис. 5.13](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.13).

  
**Рис. 5.13.**Создание нового отчета

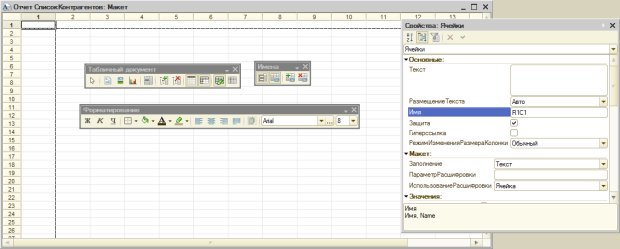
Первым этапом работы над отчетом станет создание макета отчета. *Макет* позволяет заранее определить и оформить "блоки", из которых будет построен отчет.

Следует отметить, что во всех возможных случаях при разработке *прикладных решений* для 1С:Предприятие 8.2. следует создавать их на основе схемы компоновки данных. Однако умение работать с макетами в форме табличных документов может пригодиться в том случае, если вам понадобится отредактировать сторонний отчет, выполненный в таком стиле.

Перейдем на закладку формы редактирования объекта **Макеты** и нажмем на кнопку **Добавить**. Появится окно конструктора макета, где нам предложат задать его имя (оставим имя *по* умолчанию – **Макет**), и тип макета – нас устроит **Табличный документ**, [рис. 5.14](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.14).

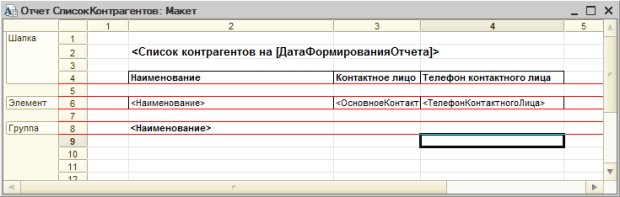
  
**Рис. 5.14.**Создание макета для отчета

После нажатия на кнопку **Готово**, мы видим табличный редактор, [рис. 5.15](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.15)., очень напоминающий Microsoft *Excel*. Работая с ним, мы можем пользоваться стандартной палитрой свойств, а так же – панелями инструментов, в частности – **Форматирование**, **Табличный документ**, **Имена**. Наша задача сейчас – создать и отформатировать области, которые позже будут использованы для формирования готового отчета.

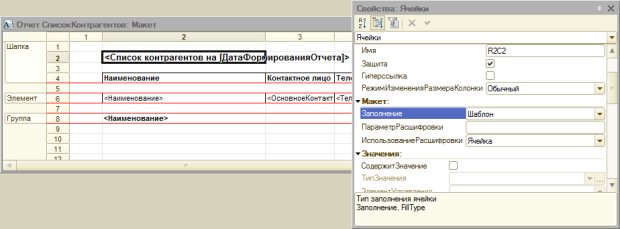
[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_15.jpg)  
**Рис. 5.15.**Средства редактирования макета отчета

При создании макета мы можем вводить в ячейки обычный текст – такой текст отображается в ячейке без каких-либо дополнительных знаков. *Ячейка* может содержать именованный *параметр*, который будет заполнен при формировании отчета. Так же ячейки могут содержать шаблоны, состоящие из обычного текста и параметров, которые так же можно заполнить.

На [рис. 5.16](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.16). показан готовый *макет*.

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_16.jpg)  
**Рис. 5.16.**Готовый макет отчета

*Ячейка* 2,2 заполнена следующим образом: в нее сначала введен текст **"Список контрагентов на [ДатаФормированияОтчета]"**, после чего вызвано окно свойств этой ячейки, в которых, в свойстве **Заполнение** выбрано **Шаблон**, [рис. 5.17](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.17).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_17.jpg)  
**Рис. 5.17.**Настройка ячейки, содержащей шаблон

*Параметр* **ДатаФормированияОтчета** мы установим в текущую дату программно при формировании отчета.

Ячейки с 4,2 *по* 4,4 содержат обычный текст – он будет выводиться в качестве шапки таблицы.

И заголовок отчета и шапка таблицы объединены в область с именем **Шапка**. Для задания имени области достаточно выделить нужные ячейки (выделять нужно *по* заголовкам строк) и отредактировать в палитре свойств *параметр* **Имя выделенного диапазона**, или воспользоваться кнопкой **Назначить имя**панели инструментов **Имена**.

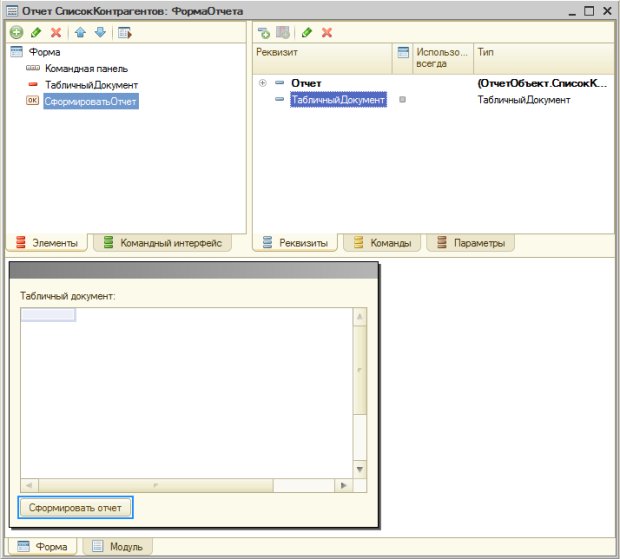
Область **Элемент** содержит три параметра – **Наименование**, **ОсновноеКонтактноеЛицо** и **ТелефонКонтактногоЛица**. После ввода в каждую из ячеек имен параметров, нужно выделить их (все вместе или *по* одной) и в окне свойств в *поле* **Заполнение** указать **Параметр**. К тексту в ячейках будут автоматически добавлены угловые скобки (<>), что позволяет визуально определить наличие в ячейке параметра.

Область **Группа** содержит лишь *параметр* **Наименование**.

Обратите внимание на имена параметров – они соответствуют именам реквизитов справочника, которыми мы собираемся их заполнять.

Ячейки в шаблоне можно форматировать – задавать их границы, оформление текста, выравнивание и т.д.

Теперь приступим к созданию формы отчета. Перейдем на вкладку **Формы** окна редактирования объекта, добавим новую форму отчета, оставим все настройки в состоянии *по* умолчанию и нажмем **Готово**. Добавим, на вкладке **Реквизиты** редактора форм новый *реквизит*, назовем его **ТабличныйДокумент**, выберем для него тип **ТабличныйДокумент**. Перетащим созданный *реквизит* в *поле***Элементы**. В состав команд формы добавим новую команду, зададим ей имя **СформироватьОтчет** и так же переместим в *поле***Элементы**. В итоге у нас получится форма, выглядящая так, как показано на [рис. 5.18](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=4#image.5.18).

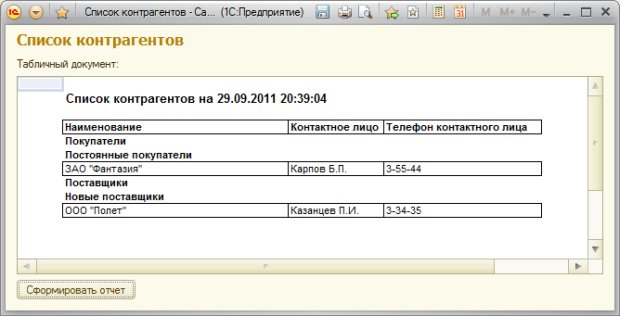
  
**Рис. 5.18.**Настройка формы отчета

Для построения отчета мы должны будем получать данные из справочника. Это можно сделать с помощью уже знакомого вам объекта **СправочникВыборка**, можно получить данные с помощью запроса. В любом случае, это предусматривает работу с базой данных, то есть, нам понадобится процедура, выполняемая на сервере. До настоящего времени мы пользовались лишь директивой компиляции **&НаСервере** – при вызове методов, объявленных с этой директивой, мы имеем *доступ* к контексту формы, при этом между клиентом и сервером происходит передача дополнительных данных – как при вызове серверного метода с клиента на *сервер*, так и в обратном направлении. Сейчас мы воспользуемся серверным внеконтекстным методом для формирования отчета. В этом методе мы собираемся формировать табличный документ, содержащий данные отчета.

При реализации метода в виде процедуры, нам придется передать в него в качестве параметра наш *реквизит* **ТабличныйДокумент**. *По* умолчанию параметры передаются *по* ссылке, то есть, работать процедура будет непосредственно с нашим реквизитом.

При реализации метода в виде функции мы можем ничего не передавать в него, сформировать внутри функции табличный документ и вернуть уже заполненный документ в точку вызова, присвоив его нашему реквизиту **ТабличныйДокумент**.

Реализуем метод в виде функции. Готовый код формирования отчета ([рис. 5.19](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=5#image.5.19)) будет выглядеть следующим образом:



**Рис. 5.19.**Готовый отчет

&НаКлиенте

Процедура СформироватьОтчет(Команда)

ТабличныйДокумент=СформироватьОтчетНаСервере();

КонецПроцедуры

&НаСервереБезКонтекста

Функция СформироватьОтчетНаСервере()

ТабличныйДокумент=Новый ТабличныйДокумент();

Макет=Отчеты.СписокКонтрагентов.ПолучитьМакет("Макет");

Шапка=Макет.ПолучитьОбласть("Шапка");

Элемент=Макет.ПолучитьОбласть("Элемент");

Группа=Макет.ПолучитьОбласть("Группа");

Шапка.Параметры.ДатаФормированияОтчета=ТекущаяДата();

ТабличныйДокумент.Вывести(Шапка);

Выборка=Справочники.Контрагенты.ВыбратьИерархически();

Пока Выборка.Следующий() Цикл

Если Выборка.ЭтоГруппа Тогда

Область=Группа;

Иначе

Область=Элемент;

КонецЕсли;

Область.Параметры.Заполнить(Выборка);

ТабличныйДокумент.Вывести(Область);

КонецЦикла;

Возврат (ТабличныйДокумент);

КонецФункции

В клиентской процедуре **СформироватьОтчет()**мы вызываем серверную внеконтекстную функцию **СформироватьОтчетНаСервере()**, присваивая возвращаемое ей *значение* реквизиту **ТабличныйДокумент**.

В серверной функции мы создаем новую переменную с типом **ТабличныйДокумент** и именем **ТабличныйДокумент**. Именно в него мы будем выводить данные и именно его будем возвращать в точку вызова. Несмотря на то, что имя реквизита формы и имя данной переменной совпадают, между ними нет никакой связи. Это – отдельные объекты.

Далее, мы получаем *макет* из нашего отчета, пользуясь методом **ПолучитьМакет()**и задавая имя макета. Отчет может иметь несколько макетов, их выбор осуществляется *по* имени.

В переменную **Шапка** мы записываем область макета **Шапка**, соответственно поступаем с переменными **Элемент** и **Группа**.

Командой **Шапка.Параметры.ДатаФормированияОтчета=ТекущаяДата();** мы заполняем ранее заданный в макете *параметр***ДатаФормированияОтчета**, записав в него текущую дату. Дата, возвращаемая функцией **ТекущаяДата**, содержит, помимо года, месяца и дня, так же часы, минуты и секунды. При необходимости дату перед выводом можно отформатировать при помощи функции **Формат()**.

После того, как *параметр*, находящийся в шапке, заполнен, мы можем вывести шапку в табличный документ командой **ТабличныйДокумент.Вывести(Шапка);**

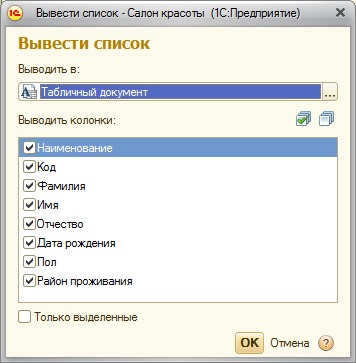
Следующим этапом нашей работы будет получение иерархической выборки справочника. Такая *выборка* позволяет получить элементы и группы справочника в иерархическом порядке, учитывая родительские отношения между элементами.

В цикле обхода выборки мы сначала проверяем, является ли текущий элемент справочника группой. Если является, присваиваем переменной **Область** ранее полученную область отчета **Группа**. Если не является – присваиваем ей область **Элемент**.

Благодаря совпадению имен параметров и имен реквизитов справочника, для помещения данных из выборки в область макета, достаточно воспользоваться конструкцией **Область.Параметры.Заполнить(Выборка);**. После заполнения параметров мы выводим сформированную область в табличный документ.

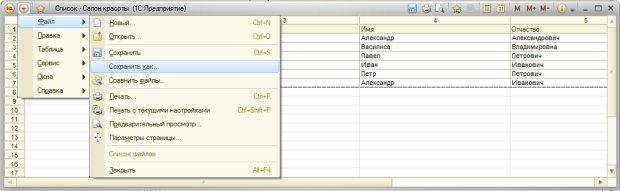
Когда цикл перебора выборки будет завершен, мы возвращаем сформированный табличный документ в точку вызова и *пользователь* видит готовый отчет.

Для формирования простейших отчетов, *пользователь* может воспользоваться стандартной функциональностью, присутствующей в 1С:Предприятие 8. Для этого, открыв, например, *список* справочника, он может выполнить команду **Все действия > Вывести список**. Появится окно **Вывести список**, [рис. 5.20](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=5#image.5.20), где в *поле* **Выводить в** можно выбрать либо **Табличный документ** (его обычно и используют), либо – **Текстовый документ.**



**Рис. 5.20.**Окно Настройка списка

В *поле* **Выводить колонки** можно настроить состав выводимых в документ колонок (в нашем случае *команда* выполнена для справочника **ФизическиеЛица**). После нажатия на **ОК** выбранные данные оформляются в виде табличного документа, а с помощью команды **Файл > Сохранить как**, [рис. 5.21](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=5#image.5.21)., этот документ можно сохранить в нужном формате для дальнейшей обработки в других приложениях.

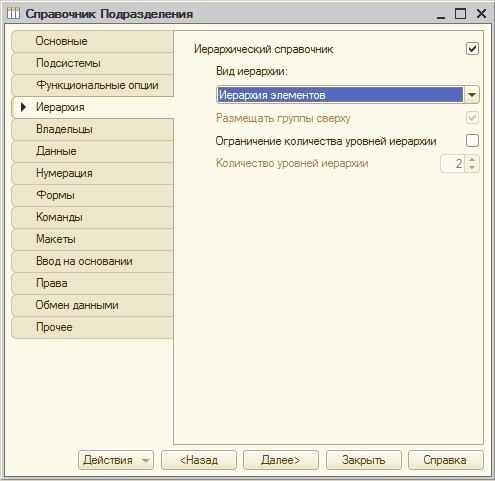
[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_21.jpg)

**Рис. 5.21.**Вывод данных в табличный документ

### Справочник с иерархией элементов

Добавим в нашу конфигурацию еще один справочник. Дадим ему имя **Подразделения**, добавим в состав подсистем **БухгалтерскийУчет**, **УчетРаботыМастеров** и **РасчетЗаработнойПлаты**. Увеличим длину наименования на закладке **Данные** до 100 символов. Сделаем справочник иерархическим – на закладке **Иерархия** установим флаг **Иерархический справочник**, *параметр* **Вид иерархии** установим в *значение* **Иерархия элементов**, [рис. 5.22](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=6#image.5.22).

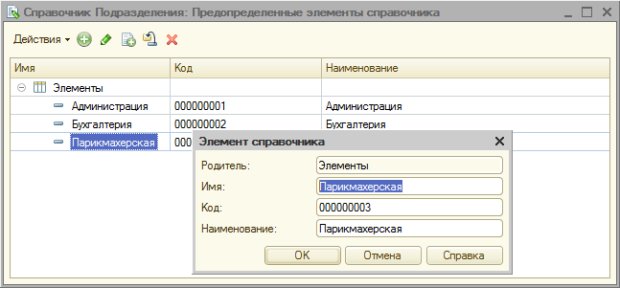
*Иерархия* элементов вполне логична для справочника **Подразделения**, так как одни *подразделения* могу включать в себя другие, и, при этом, вполне самостоятельны, их можно выбирать при заполнении, например, реквизитов других справочников, в то время, как при иерархии групп и элементов, группы играют лишь вспомогательную роль для организации информации внутри справочника.

  
**Рис. 5.22.**Настройка иерархии справочника Подразделения

Кроме того, в справочник **Подразделения** мы добавим несколько предопределенных элементов. Эти элементы справочника задаются в **Конфигураторе**, *пользователь* обладает лишь ограниченными возможностями *по* управлению ими, в частности, не может их удалить. Такие элементы обычно создают для того, чтобы ими можно было удобно и надежно оперировать в программном коде, не опасаясь того, что *пользователь* удалит их.

Для этого перейдем на вкладку окна редактирования объекта **Прочее** и нажмем на вкладку **Предопределенные**. В окне ввода предопределенных элементов справочника введем следующие ([рис. 5.23](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=6#image.5.23).):

* Администрация
* Бухгалтерия
* Парикмахерская



**Рис. 5.23.**Создание предопределенных элементов справочника Подразделения

Доработаем справочник **Сотрудники**. Снабдим его следующими реквизитами, [рис. 5.24](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=6#image.5.24).:

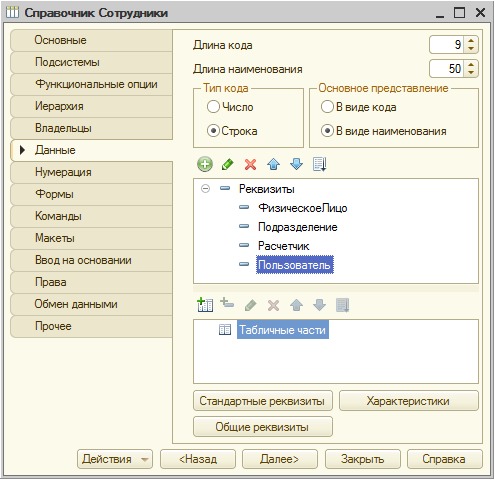
**Имя**: ФизическоеЛицо, **Тип**: СправочникСсылка.ФизическиеЛица

**Имя**: Подразделение, **Тип**: СправочникСсылка.*Подразделения*

**Имя**: Расчетчик, **Тип**: Булево

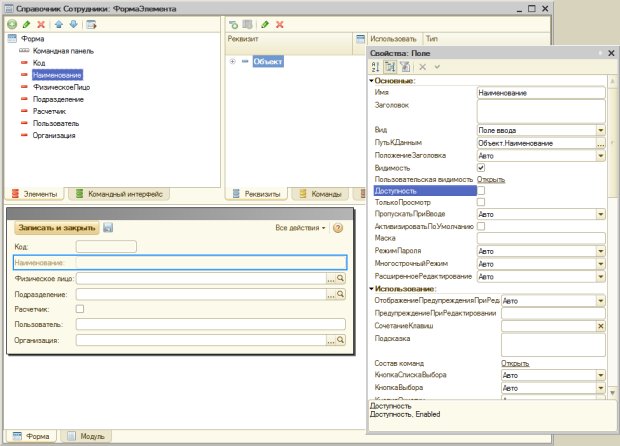
**Имя**: *Пользователь*, **Тип**: Строка, *длина* 50.

Увеличим длину **наименования** до 50 символов.



**Рис. 5.24.**Реквизиты справочника Сотрудники

Мы хотели бы, чтобы наименование сотрудника в данном справочнике формировалось бы автоматически и состояло бы из ФИО физического лица и *подразделения*, в котором работает сотрудник. Создадим форму элемента справочника и, для элемента формы **Наименование**, снимем флаг **Доступность**, [рис. 5.25](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=6#image.5.25).

[](https://www.intuit.ru/EDI/13_12_15_4/1449958784-19792/tutorial/1046/objects/5/files/05_25.jpg)  
**Рис. 5.25.**Настройка формы элемента справочника Сотрудники

Теперь подумаем над тем, как автоматически заполнить *поле* **Наименование** на основе данных полей **Физическое лицо** и **Подразделение**. Сделать это можно различными способами, мы реализуем следующую функциональность: перехватим события изменения полей **Физическое лицо** и **Подразделение** и вызовем в обработчике каждого из этих событий процедуру, заполняющую *поле* **Наименование**. Так *пользователь*, заполняющий элемент справочника, сможет сразу же увидеть результаты формирования наименования.

Нашей задаче отвечает следующий код:

&НаКлиенте

Процедура ФизическоеЛицоПриИзменении(Элемент)

СформироватьНаименование();

КонецПроцедуры

&НаКлиенте

Процедура ПодразделениеПриИзменении(Элемент)

СформироватьНаименование();

КонецПроцедуры

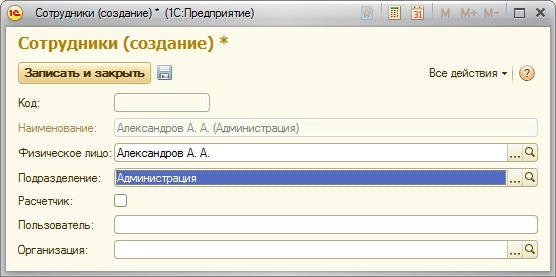
Процедура СформироватьНаименование()

Объект.Наименование=Объект.ФизическоеЛицо.Наименование +" ("+

Объект.Подразделение.Наименование+") ";

КонецПроцедуры

Результаты работы созданного нами механизма показаны на [рис. 5.26](https://www.intuit.ru/studies/courses/2318/618/lecture/13472?page=6#image.5.26).



**Рис. 5.26.**Настройка формы элемента справочника Сотрудники