

1С® БИБЛИОТЕКА РАЗРАБОТЧИКА

Е. Ю. Хрусталева

101 СОВЕТ

**НАЧИНАЮЩИМ РАЗРАБОТЧИКАМ
В СИСТЕМЕ «1С:ПРЕДПРИЯТИЕ 8»**



1С®
ПАБЛИШИНГ

Хрусталева Е.Ю.

101 совет начинающим разработчикам в системе "1С:Предприятие"

Электронная книга в формате pdf; ISBN 978-5-9677-2348-3.

Электронный аналог издания "101 совет начинающим разработчикам в системе «1С:Предприятие 8»" (ISBN 978-5-9677-2322-3, М.: ООО "1С-Паблишинг", 2015; артикул печатной книги по прайс-листу фирмы "1С": 4601546117618; по вопросам приобретения печатных изданий издательства "1С-Паблишинг" обращайтесь к партнеру "1С", обслуживающему вашу организацию, или к другим партнерам фирмы "1С").

Книга адресована начинающим разработчикам прикладных решений в системе "1С:Предприятие 8". Она поможет овладеть наиболее эффективными приемами разработки приложений.

Средства разработки "1С:Предприятия 8" позволяют решать широкий круг задач. Начинающим разработчикам бывает сложно сориентироваться в многообразии имеющихся инструментов и возможностей. Зачастую бывает так, что одна и та же задача может быть выполнена с помощью разных инструментов или разными способами. Но непонятно, какой из этих способов является предпочтительным. Именно в таких случаях поможет эта книга. Она содержит набор простых примеров, показывающих один наиболее эффективный способ решения той или иной задачи, возникающей в процессе разработки.

Примеры достаточно подробно описаны и проиллюстрированы для того, чтобы их мог воспроизвести начинающий. В то же время за более подробной информацией об используемых инструментах следует обращаться к документации "1С:Предприятию 8".

Для создания демонстрационных примеров использовалась версия 8.3.5.1443 платформы "1С:Предприятие 8".

Книга выпущена под редакцией Максима Радченко.

Интернет-конференция для начинающих разработчиков
<http://devtrainingforum.v8.1c.ru/forum>.

Оглавление

Введение	7
Удобное рабочее пространство.....	9
Пример 1. Раскрыть окно на весь экран.....	10
Пример 2. Свернуть окна, которые не нужны постоянно.....	12
Пример 3. Скрыть лишние панели	13
Пример 4. Подсвечивать парные синтаксические конструкции	15
Пример 5. Автоматически подсвечивать текущие идентификаторы	17
Пример 6. Подсвечивать выбранные идентификаторы.....	18
Пример 7. Отобразить непечатаемые символы	20
Пример 8. Закрыть сразу все открытые окна.....	22
Видеть только нужные объекты.....	23
Пример 9. Показать только те объекты, которые относятся к выбранной подсистеме	24
Пример 10. Показать объекты из нескольких подсистем.....	25
Пример 11. Палитра свойств в виде закладок	26
Пример 12. Показать синтакс-помощник для мобильного приложения	27
Пример 13. Показать синтакс-помощник на двух языках	28
Пример 14. Скрыть содержимое циклов и условий.....	30
Пример 15. Объединить несколько процедур в группу	33
Получение помощи и подсказок.....	37
Пример 16. Подсказки при наборе текста программы	38
Пример 17. Почему контекстная подсказка подсказывает не всегда	38

Пример 18. Быстро открыть описание метода в синтакс-помощнике	40
Пример 19. Найти открытое описание в дереве синтакс-помощника.....	41
Пример 20. Быстро найти в синтакс-помощнике описание свойства из палитры свойств	42
Пример 21. Что делать, когда описания в синтакс-помощнике недостаточно.....	44
Пример 22. Произвольный поиск в синтакс-помощнике	46
Найти что-то.....	49
Пример 23. Как понять что искать	50
Пример 24. Самый простой способ поиска объектов.....	51
Пример 25. Найти объект по имени в небольшой конфигурации	53
Пример 26. Найти объект по синониму или представлению в небольшой конфигурации	54
Пример 27. Найти объект по представлению в крупной конфигурации	57
Пример 28. Найти объект по алфавиту	59
Пример 29. Самый простой способ поиска свойств	60
Пример 30. Найти свойство по алфавиту.....	63
Пример 31. Подсвечивать найденные фрагменты в тексте программы	64
Пример 32. Найти свойство или метод в синтакс-помощнике.....	66
Пример 33. Что написано в сообщении об ошибке	68
Пример 34. Быстро перейти к строке, в которой ошибка.....	69
Пример 35. Найти строку, про которую сказано в сообщении об ошибке	71
Пример 36. Найти объект конфигурации при выборе	75
Удобное редактирование	77
Пример 37. Быстро установить принадлежность к подсистемам	78
Пример 38. Оценить внешний вид формы	79
Пример 39. Перейти к известной процедуре модуля	81
Пример 40. Находясь в месте вызова процедуры, перейти к ее содержимому	83
Пример 41. Узнать, в каких модулях используется данная процедура	85
Не создавать одно и то же	87
Пример 42. Копировать объекты в небольшой конфигурации	88
Пример 43. Копировать объекты из другой конфигурации	91
Пример 44. Редактировать сразу несколько реквизитов	92
Пример 45. Копировать реквизиты, команды и элементы.....	93
Пример 46. Перетаскивать элементы	95
Пример 47. Перетаскивать имена объектов в код модуля	97
Пример 48. Перетаскивать методы, конструкторы в код модуля.....	98
Пример 49. Создать копию базы для экспериментов.....	100
Пользоваться автоматизацией там, где это возможно	105
Пример 50. Создать поле ввода.....	108

Пример 51. Создать поле табличного документа.....	109
Пример 52. Создать поле флажка	110
Пример 53. Создать поле картинки.....	111
Пример 54. Создать поле переключателя.....	112
Пример 55. Создать поле HTML-документа.....	113
Пример 56. Удалить элемент формы или команду	114
Пример 57. Создать кнопку.....	115
Пример 58. Создать обработчик события	119
Пример 59. Удалить обработчик события	122
Пример 60. Подсказать пользователю назначение элемента.....	124
Пример 61. Не писать «вручную» имена переменных, свойств и методов.....	128
Пример 62. Настройте контекстную подсказку.....	130
Пример 63. Подсказка после знака равенства.....	130
Пример 64. Подсказка после точки	131
Пример 65. Изменять имена с помощью подсказки	132
Пример 66. Подсказка при написании параметров	134
Пример 67. Подсказка после кавычки.....	135
Пример 68. Подсказка после оператора «Новый»	138
Пример 69. Настроить шаблоны текста	139
Пример 70. Автоматически подставлять шаблоны	140
Пример 71. Создать собственные шаблоны	143
Пример 72. Создать форматную строку	146
Пример 73. Использовать синтаксический отступ	149
Пример 74. Форматировать текст модуля	150
Пример 75. Закомментировать фрагмент программы	152
Пример 76. Переносить длинные строки.....	152
Пример 77. Создать текст запроса	153
Пример 78. Создать запрос и обработать его результат.....	157
Пример 79. Проверить или изменить текст запроса	161
Пример 80. Написать и отладить запрос	163
Пример 81. Автоматически проверять синтаксис.....	166
Пример 82. Как исправить ошибку	167
Пример 83. Внимательно анализировать текст	167
Пример 84. Запустить отладочный сеанс от имени другого пользователя.....	169
Пример 85. Подключить отладчик к работающему сеансу	171
Пример 86. Узнать значения переменных.....	173
Пример 87. Остановить исполнение до того, как произойдет ошибка.....	178
Пример 88. Узнать значения нескольких переменных одновременно	179

Пример 89. Остановить исполнение при выполнении некоторого условия	181
Пример 90. Посмотреть изменение переменных по шагам.....	182
Пример 91. Узнать, откуда была вызвана процедура.....	186
Пример 92. Посмотреть результат выполнения запроса.....	188
Пример 93. Выделить часть программы в отдельную процедуру	189
Пример 94. Изменить имя переменной или процедуры	193
Пример 95. Создать описание процедуры	197
Пример 96. Создать обработку оповещения.....	199
Пример 97. Создать строку на разных языках.....	201
Пример 98. Изменить интерфейсные названия.....	203
Пример 99. Выбирать поля в системе компоновки	207
Пример 100. Оценить производительность программного кода	208
Пример 101. Найти неиспользуемые процедуры в коде конфигурации.....	211

Введение

Система «1С:Предприятие» содержит в своем составе среду разработки прикладных решений, которая называется *Конфигуратор*. Конфигуратор позволяет выполнять все задачи, которые могут возникнуть при разработке прикладных решений. Круг этих задач довольно широк, поэтому и конфигуратор обладает большим количеством разных инструментов: окна, панели, редакторы, конструкторы, обработки, механизмы и т. д.

Начинающему разработчику бывает тяжело разобраться во всех возможностях имеющихся инструментов. Часто разработчики выбирают самый простой путь – попытаться решить имеющуюся задачу опираясь на свою интуицию. И зачастую добиваются успеха, потому что интерфейс конфигуратора довольно дружественный и позволяет в большинстве случаев самостоятельно найти какой-то способ решения возникшей задачи.

Недостаток такого подхода заключается в том, что найденный способ далеко не всегда является самым эффективным. Скорее всего, те же действия можно выполнить быстрее или даже полностью автоматически, если использовать другие инструменты, другие способы.

Именно такие приемы разработки, которые могут значительно «облегчить жизнь» разработчикам, мы и рассмотрим в данной книге. Наверняка они будут интересны начинающим разработчикам, потому что позволят избежать многочисленных ошибок. Позволят сделать процесс разработки менее рутинным и более интересным. А опытные разработчики в процессе чтения книги могут почерпнуть советы, которые позволят разрабатывать приложения более быстро и качественно.

В этой книге мы не будем подробно рассказывать, как устроен тот или иной инструмент для разработки прикладных решений. Мы покажем лишь, какие возможности инструментов конфигуратора нужно использовать, чтобы максимально эффективно решать возникающие задачи. Если вы хотите больше узнать о том или ином инструменте, который используется в примере, то для этого вам нужно обращаться к документации по системе «1С:Предприятие 8».

Удобное рабочее пространство

Чтобы сделать разработку прикладных решений более быстрой, удобной и привлекательной, вы можете организовать для себя рабочее пространство в соответствии со своими вкусами и потребностями.

Часто при разработке требуется максимизировать рабочую область конфигуратора, чтобы лучше сосредоточиться на том, чем вы в данный момент занимаетесь. Для этого вы можете раскрыть текущее рабочее окно на весь экран, свернуть и сделать всплывающими окна, которые в данный момент не нужны, а также скрыть лишние панели инструментов конфигуратора. Кроме того, все открытые в процессе работы окна вы можете закрыть одним нажатием кнопки.

В процессе написания кода удобно сразу же контролировать себя при помощи подсветки парных синтаксических конструкций, а также текущих или выбранных идентификаторов. Кроме того, иногда полезной может быть возможность отображения в модулях непечатаемых символов. Таким образом, какая-то часть ошибок отпадет «сама по себе», и процесс разработки станет более легким и приятным.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 1. Раскрыть окно на весь экран

Предположим, вы работаете над каким-то фрагментом конфигурации, и у вас открыто несколько форм, модулей объектов и общих модулей, задействованных в отлаживаемой функциональности.

Понятно, что нагромождение окон и их лишнее прокручивание мешает сосредоточиться и отвлекает внимание.

Поэтому удобно раскрыть окно, с которым вы в данный момент работаете, на весь экран конфигуратора.

При этом если понадобится перейти в другое окно, то сделать это можно при помощи панели открытых окон, отображаемой в нижней строке конфигуратора (рис. 1).

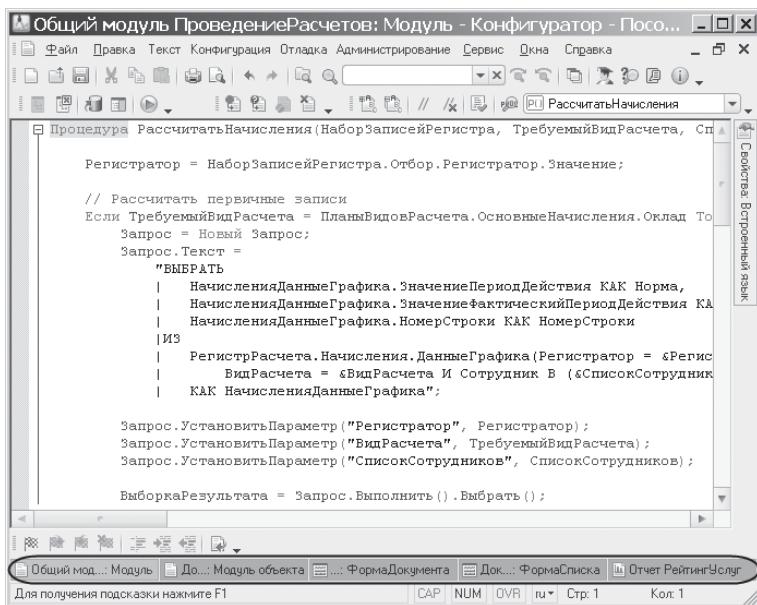


Рис. 1. Максимизация окон, открытых в конфигураторе

Для экономии пространства панель окон можно сделать прячущейся. Этим свойством, как и вообще отображением панели окон, можно управлять из контекстного меню, вызываемого на свободной области конфигуратора.

Для того чтобы панель окон отображалась в конфигураторе, но была прячущейся, в контекстном меню рабочей области конфигуратора установите флажки Панель окон и Автоматически прятать (рис. 2).

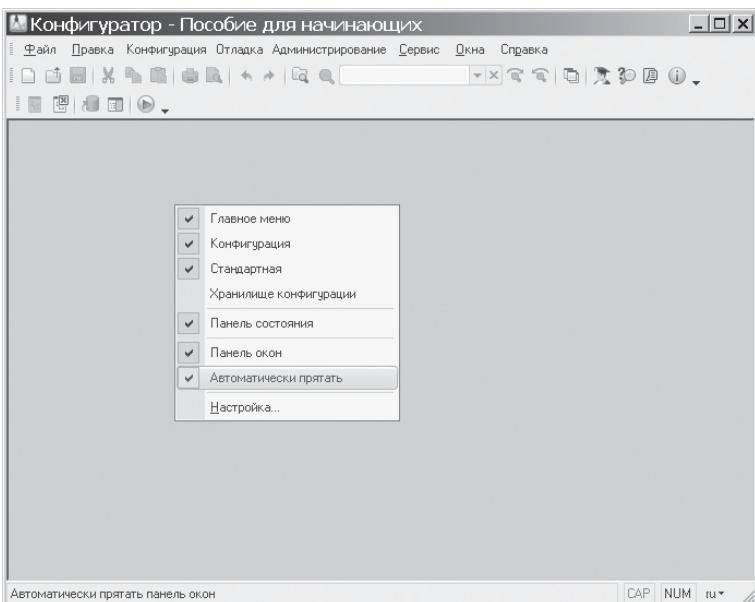


Рис. 2. Установка панелей конфигуратора из контекстного меню

Теперь, чтобы панель окон появилась, просто подведите мышь к нижней строке конфигуратора.

ПРИМЕЧАНИЕ

Если панель окон не отображается (флажок Панель окон отключен), переходить по открытым окнам конфигуратора можно из списка окон, содержащегося в подменю Окна главного меню.

Пример 2. Свернуть окна, которые не нужны постоянно

Предположим, вы работаете с какой-то формой, устанавливаете свойства элементов формы и редактируете ее модуль. В процессе работы вы все время переходите из модуля формы в палитру свойств и обратно. Окно палитры свойств может закрывать ту часть модуля, которая в данный момент вам нужна.

Поэтому приходится все время закрывать, открывать или сдвигать окно палитры. Это неудобно.

В данном случае удобно сделать это окно прячущимся, чтобы оно открывалось только в нужный момент по желанию разработчика.

Для этого вы можете нажать символ кнопки в правом углу заголовка окна палитры свойств. При этом при наведении курсора мыши на любое другое окно палитра свойств будет сворачиваться на дополнительную панель в правой части экрана (рис. 3).

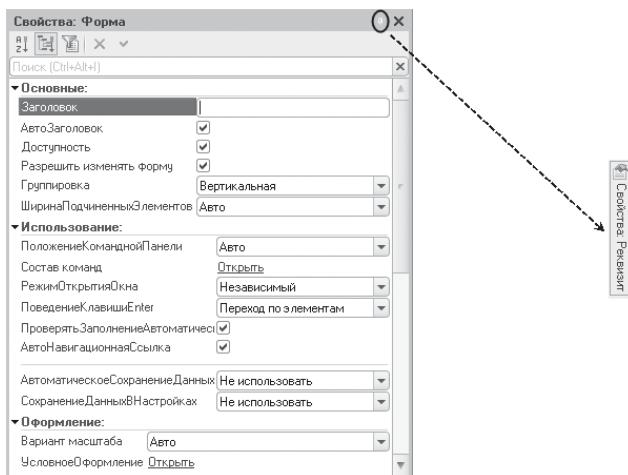


Рис. 3. Сворачивание палитры свойств

При наведении курсора мыши на изображение свернутой палитры свойств она будет открываться. Так же, как обычно, палитра свойств будет открыта при двойном щелчке на элементе, реквизите или команде формы.

Аналогичным образом можно сделать прячущимся другие системные окна – окно дерева объектов конфигурации, окно синтакс-помощника и т. п.

Пример 3. Скрыть лишние панели

Для удобства в основном окне конфигуратора расположены различные панели инструментов Стандартная, Конфигурация, Модуль и т. п., на которые помещены наиболее часто используемые кнопки, дублирующие команды главного меню.

Но далеко не всем разработчикам они действительно нужны. Некоторым они мешают тем, что отнимают часть рабочего пространства. Ведь можно пользоваться только главным меню, так как в нем есть абсолютно все команды, которые вызываются кнопками на панелях инструментов.

В этом случае вы можете отключить все панели инструментов, кроме главного меню. Для этого выполните команду главного меню Сервис > Настройка и отключите видимость всех панелей инструментов, кроме панели Главное меню (рис. 4).

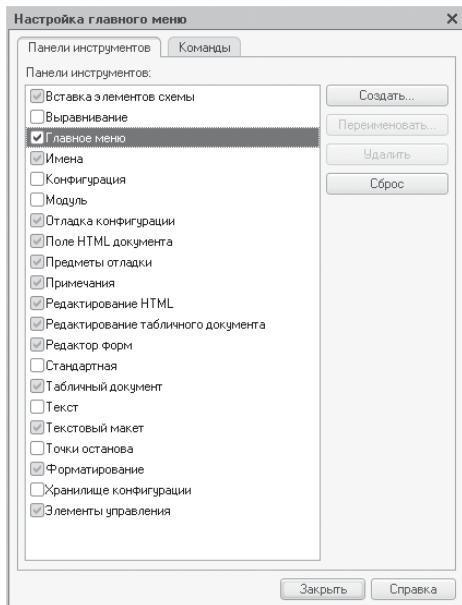


Рис. 4. Настройка видимости панелей инструментов конфигуратора

Если же вы отключили случайно все панели, включая главное меню, то так работать вы не сможете. Чтобы включить главное меню (и/или другие панели конфигуратора), закройте все открытые окна конфигуратора комбинацией клавиш **Ctrl + F4**, на пустой рабочей области вызовите контекстное меню и выполните пункт меню **Настройка** (рис. 5).

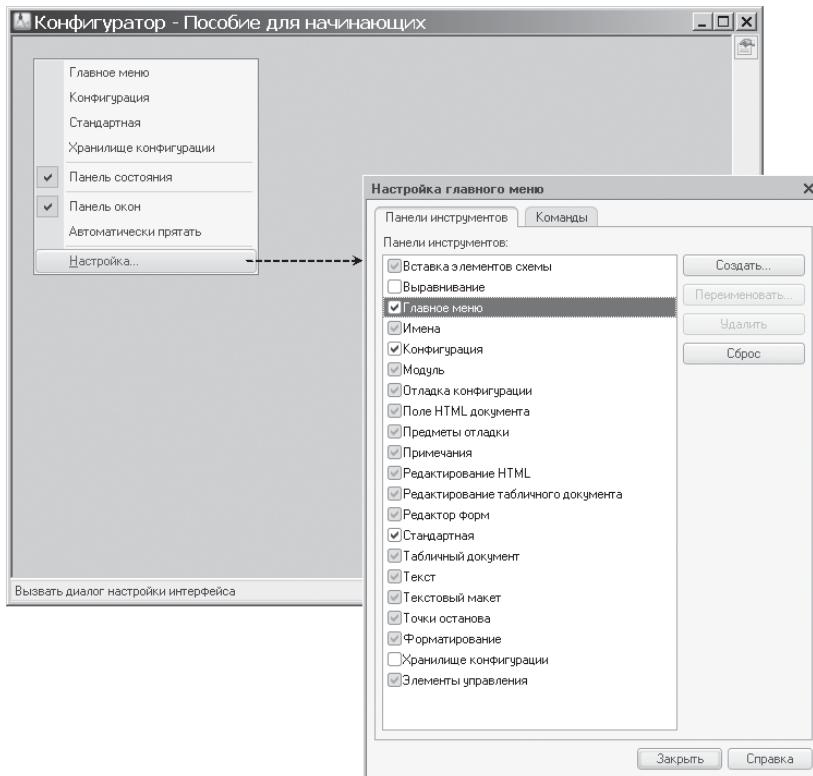


Рис. 5. Настройка видимости панелей инструментов конфигуратора

При этом за один раз можно включить сразу нескольких нужных вам панелей инструментов.

ПРИМЕЧАНИЕ

Если вы хотите отобразить только главное меню, то хорошим решением также может быть установка флашка Главное меню из контекстного меню рабочей области конфигуратора.

Пример 4. Подсвечивать парные синтаксические конструкции

Наверное, никто не обходится без ошибок при написании сложных вложенных друг в друга условий или выражений. В конце рабочего дня, когда концентрация внимания уже на нуле, легко потерять, например, закрывающую скобку при написании какого-то выражения или можно забыть пропустить закрывающую конструкцию от парных конструкций вроде: Если, Цикл, Для Каждого. В результате при синтаксической проверке модуля вы можете получить несколько связанных друг с другом ошибок и долго и нудно искать, где же вы ошиблись.

Для того чтобы не допустить такую ошибку или, если уж она случилась, помочь исправить ее, в конфигураторе есть специальные средства. Они позволяют выделять цветом границы блока (начало и конец процедуры, начало и конец цикла, начала и конец условного перехода, парные скобки) в тексте модуля.

Эта возможность включена стандартно. Например, если установить курсор на ключевое слово Процедура, будет выделено окончание этой процедуры – КонецПроцедуры. Если установить курсор на ключевое слово Если, будет выделено окончание этого условия – КонецЕсли. Если выделить открывающую скобку, будет выделена соответствующая ей закрывающая скобка (рис. 6).

Это очень удобно и полезно для самоконтроля написанного кода, еще до синтаксической проверки модуля. В процессе написания кода это удобно тем, что когда вы пишете закрывающую конструкцию, подсвечивается открывающая. То есть визуально вы видите, что оператор (или скобки) написаны правильно. А при исправлении подобных ошибок подсветка удобна для того, чтобы быстро найти конструкции, у которых нет «пары».

Если по какой-то причине подсветка парных синтаксических конструкций у вас отключена, то включить ее можно в параметрах конфигуратора: Сервис > Параметры > Модули > Редактирование, окно Выделение цветом синтаксических конструкций > Границы блока. Там же вы можете восстановить стандартное выделение и других элементов. Например, обычно ключевые слова выделяются красным цветом текста, идентификаторы – синим, комментарии – зеленым и т. д.

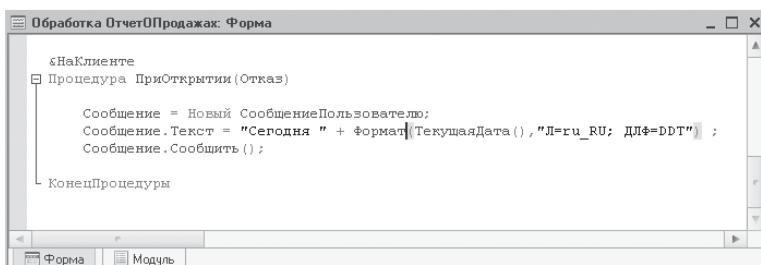
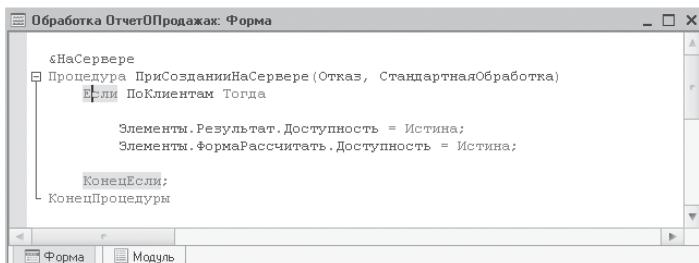
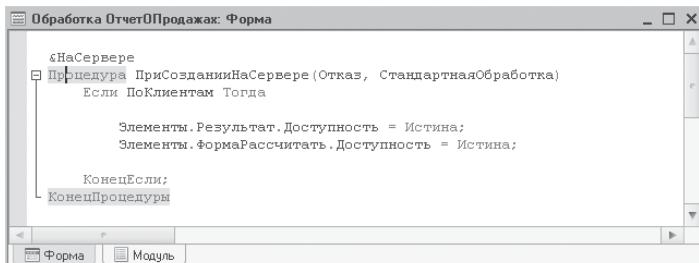


Рис. 6. Подсветка синтаксических конструкций в модуле

ПРИМЕЧАНИЕ

Стандартно границы блока в модуле выделяются светлым серо-коричневым цветом фона. Если для вас это недостаточно ярко, то можно выделить границы блока другим, более заметным цветом фона.

Пример 5. Автоматически подсвечивать текущие идентификаторы

Предположим, вы набираете вручную, без использования контекстной подсказки, имя переменной, которую вы уже использовали ранее.

Для контроля правильности ее написания удобно, когда в модулях настроена подсветка текущих идентификаторов, которые находятся в данный момент «под курсором». В этом случае можно легко визуально увидеть, совпадает имя переменной с тем, что вы уже набрали выше, или нет.

При стандартных настройках текущие идентификаторы не подсвечиваются. Но вы можете изменить это в параметрах конфигуратора (Сервис > Параметры > Модули > Редактирование) и выделить, например, текущие идентификаторы зеленым цветом фона (рис. 7).

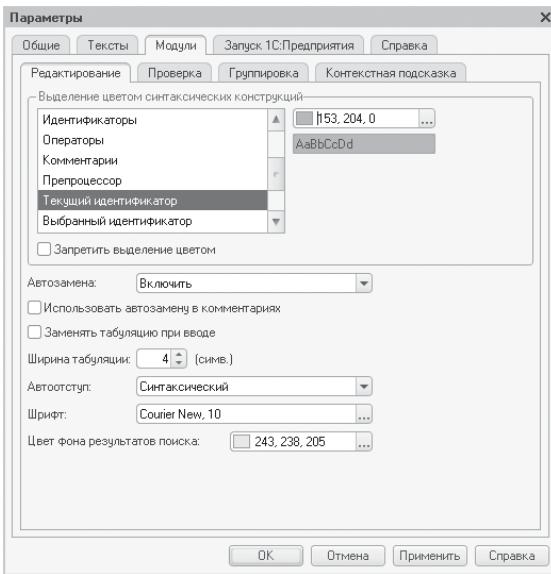


Рис. 7. Настройка подсветки текущих идентификаторов

Теперь, например, в обработке проведения документа создайте менеджер временных таблиц и сохраните его в переменной МенеджерВТ. Ниже в тексте модуля используйте эту переменную еще раз. Если вы написали имя переменной правильно, то переменная, написанная выше, будет подсвечена зеленым цветом так же, как и ваша текущая переменная (рис. 8).

```

Процедура ОбработкаПроведения(Отказ, Режим)
{
    Движения.ОстаткиМатериалов.Записывать = Истина;
    Движения.СтойкостьМатериалов.Записывать = Истина;
    Движения.Продажи.Записывать = Истина;
    Движения.Управленческий.Записывать = Истина;

    // Создать менеджер временных таблиц
    МенеджерВТ = Новый МенеджерВременныхТаблиц;
    Запрос = Новый Запрос;

    // Укажем, какой менеджер временных таблиц использует этот запрос
    Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

    Запрос.Текст =
        "ВЫБРАТЬ
        | ОказаниеУслугИПереченьНоменклатуры.Номенклатура,
        | ОказаниеУслугИПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК ВидНоменклатуры,
        | ОказаниеУслугИПереченьНоменклатуры.НаборСвойств,
        | СУММА(ОказаниеУслугИПереченьНоменклатуры.Количество) КАК КоличествоВДокументе,
        | СУММА(ОказаниеУслугИПереченьНоменклатуры.Сумма) КАК СуммаВДокументе
        | ПОМЕСТИТЬ НоменклатуроДокумента
        | ИЗ
"
}

```

Рис. 8. Выделение цветом текущих идентификаторов в тексте модуля

Если ранее написанное не подсвечивается, то, значит, в данном месте вы ошиблись.

Пример 6. Подсвечивать выбранные идентификаторы

Предположим, вы хотите быстро найти все места модуля, где содержится идентификатор, на который вы смотрите в данный момент.

В этом случае удобно, чтобы в модулях была настроена подсветка выбранных идентификаторов. Выбрав находящийся перед вами идентификатор двойным щелчком мыши, можно затем быстро пролистать модуль и найти нужное место с этим идентификатором.

При стандартных настройках выбранные идентификаторы не подсвечиваются. Но вы можете изменить это в параметрах конфигуратора (Сервис > Параметры > Модули > Редактирование) и, например, выделить выбранные идентификаторы желтым цветом фона (рис. 9).

Теперь, выделив какой-то идентификатор двойным щелчком мыши, вы можете легко увидеть все места в тексте модуля, где используется данный идентификатор.

Например, выделив идентификатор Движения в процедуре проведения документа, вы можете быстро найти все места в модуле, где создаются и изменяются движения документа (рис. 10).

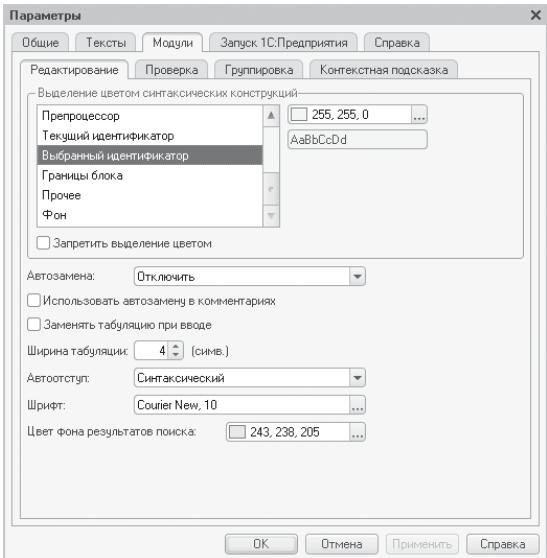


Рис. 9. Настройка подсветки выбранных идентификаторов

```
Документ ПриходнаяНакладная: Модуль объекта

Процедура ОбработкаПроведения(Отказ, Режим)
    Движения.ОстаткиМатериалов.Записывать = Истина;
    Движения.СтоимостьМатериалов.Записывать = Истина;
    Движения.Управленический.Записывать = Истина;

    Для Каждого ТекСтроМатериалы Из Материалы Цикл
        // регистр ОстаткиМатериалов Приход
        Движение = Движения.ОстаткиМатериалов.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Период = Дата;
        Движение.Материал = ТекСтроМатериалы.Материал;
        Движение.НаборСвойств = ТекСтроМатериалы.НаборСвойств;
        Движение.Склад = Склад;
        Движение.Количество = ТекСтроМатериалы.Количество;

        // регистр СтоимостьМатериалов Приход
        Движение = Движения.СтоимостьМатериалов.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Приход;
        Движение.Период = Дата;
        Движение.Материал = ТекСтроМатериалы.Материал;
        Движение.Стоимость = ТекСтроМатериалы.Сумма;

        // Регистр Управленический
        Движение = Движения.Управленический.Добавить();
        Движение.СчетDr = ПланыСчетов.Основной.Товары;
        Движение.СчетKt = ПланыСчетов.Основной.РасчетыСПоставщиками;
        Движение.Период = Дата;
```

Рис. 10. Выделение цветом выбранных идентификаторов в тексте модуля

ПРИМЕЧАНИЕ

Надо заметить, что если у вас включена подсветка текущих идентификаторов, то смысл подсветки выбранных пропадает. То есть используется либо одна, либо другая настройка, одновременно использовать обе настройки нет смысла.

Если вам не нравится, когда при простом перемещении курсора по тексту он вдруг начинает подсвечиваться, то есть работает подсветка текущих идентификаторов, то нужно ее отключить. Для этого вы можете указать, что текущие идентификаторы должны подсвечиваться фоном редактирования. А затем настроить подсветку только выбранных идентификаторов. В результате идентификатор будет подсвечиваться только тогда, когда вы специально выделите его, например, двойным щелчком.

Пример 7. Отобразить непечатаемые символы

Иногда бывает нужно, чтобы в тексте модуля отображались невидимые на печати символы, такие как пробелы и табуляция.

Например, вы собираетесь скопировать текст какой-то процедуры в интернет-конференцию для разработчиков. В модуле вы видите текст, отформатированный с использованием синтаксического отступа, как вам и нужно. А после копирования в сообщение конференции текст «плывет». И вы не понимаете, почему?

В этом случае полезно включить отображение непечатаемых символов (пробелов и табуляции) и посмотреть, в чем дело. Возможно, табуляция в некоторых местах заменена пробелами. В модуле это не видно, на первый взгляд все хорошо. Но в других редакторах текста шаг табуляции может быть другой, отсюда проблема.

Чтобы увидеть непечатаемые символы в текстах модулей, установите флажок Отображать пробелы и табуляции в настройках параметров конфигуратора на закладке Тексты (рис. 11).

При этом в тексте модуля стандартно пробелы будут отображаться в виде точки, а символы табуляции в виде двойной стрелки (рис. 12).

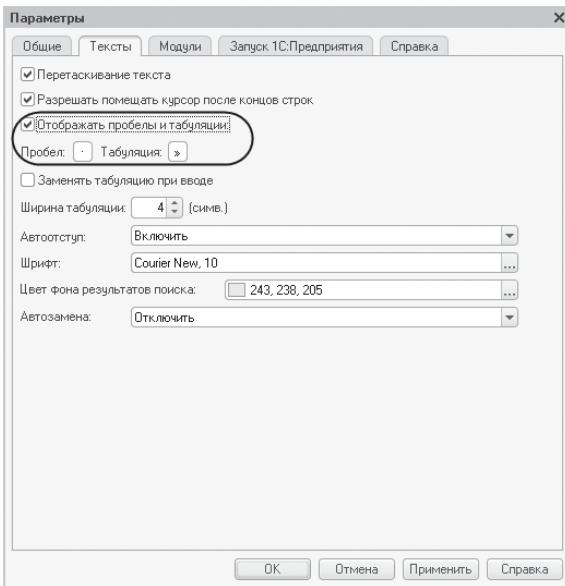


Рис. 11. Настройка параметров конфигуратора

```
сНаклиенте
Процедура ·Добавление Завершение (Результат, ·СписокНоменклатуры) ·Экспорт
    » Если ·Результат = ·КодВозврата Диалога. Да ·Тогда
    »   » ОтветПередДобавлением = ·Истина;
    »   »
    »   » Для ·Каждого ·ВыбранныйЭлемент ·Из ·СписокНоменклатуры ·Цикл
    »   »   » НоваяСтрока = ·Объект. Материалы. Добавить ();
    »   »   » НоваяСтрока. Материал = ·ВыбранныйЭлемент;
    »   » КонецЦикла;
    »   »
    » КонецЕсли; ···
КонецПроцедуры
```

Рис. 12. Отображение непечатаемых символов в тексте модуля

Пример 8. Закрыть сразу все открытые окна

Предположим, в процессе работы с конфигурацией вы открыли много различных окон форм, отчетов, общих модулей и т.п. И теперь вы хотите заняться чем-то другим, например проверкой конфигурации. А множество открытых окон на экране вам только мешает, и вы хотите их закрыть.

Закрывать каждое окно по отдельности – долго и неудобно.

Гораздо удобнее закрыть сразу все окна с помощью диалога Окна, в котором показывается список всех окон, открытых во время работы в конфигураторе.

Для этого нажмите кнопку Список окон  на панели инструментов конфигуратора, выделите в списке окна все окна с помощью сочетания клавиш Ctrl + A и нажмите кнопку Закрыть окна (рис. 13).

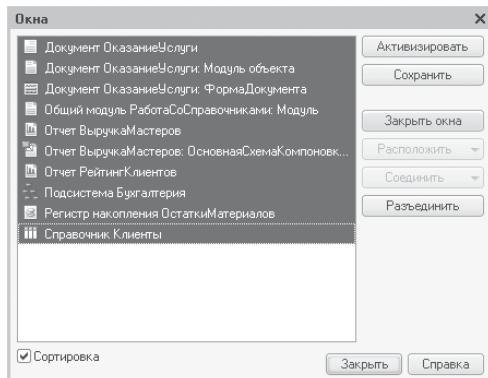


Рис. 13. Закрытие выделенных окон в диалоге «Окна»

Видеть только нужные объекты

Для быстрой и эффективной разработки прикладных решений очень удобно видеть не все, что есть в конфигурации, а только то, что нужно для конкретной решаемой в данный момент задачи.

Часто из большого числа объектов в дереве конфигурации вам бывает нужно видеть только те объекты, которые относятся к определенной функциональности. А остальные объекты, чтобы они «не мешали», нужно пока скрыть. Для этого вы можете отобрать объекты в дереве конфигурации по принадлежности к одной или сразу нескольким подсистемам.

В палитре свойств также содержится большое количество свойств объектов, которые сгруппированы по различным категориям. Чтобы видеть только ту группу свойств, которая вам в данный момент нужна, удобно представить палитру свойств в виде закладок, на каждой из которых будут размещены свойства из отдельной категории.

Также весьма полезно отображать в синтакс-помощнике только те объекты, которые доступны только в определенном контексте исполнения. Это позволит избежать ошибок при проверке доступности объектов во время синтаксического контроля модулей.

И, наконец, в конфигураторе существует удобная возможность группировки текстов модулей, которая помогает значительно упростить анализ и поиск в длинных модулях. При этом чтобы найти какое-то место в модуле, вам не нужно просматривать весь текст подряд, а только заголовки основных синтаксических конструкций или областей модуля. Дойдя до нужной группы (процедуры, цикла, области и т. п.), вы можете раскрыть ее и просмотреть ее содержимое.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 9. Показать только те объекты, которые относятся к выбранной подсистеме

Предположим, вы дорабатываете уже существующую конфигурацию. Ваша задача – изменить функциональность прикладного решения, связанную с бухгалтерией. Для более удобной и эффективной работы из большого числа объектов конфигурации вам необходимо сосредоточиться на тех объектах, которые отвечают за бухгалтерские расчеты.

Для этого в дереве конфигурации вы можете отобразить не все объекты, а только те из них, которые связаны с бухгалтерией. То есть вы можете установить фильтр объектов конфигурации по подсистеме Бухгалтерия.

Наиболее удобно и просто сделать это прямо из контекстного меню этой подсистемы.

Выделите в дереве конфигурации подсистему Бухгалтерия, вызовите ее контекстное меню и выполните команду Объекты подсистемы. В результате будут показаны только те объекты, которые принадлежат данной подсистеме (рис. 14).

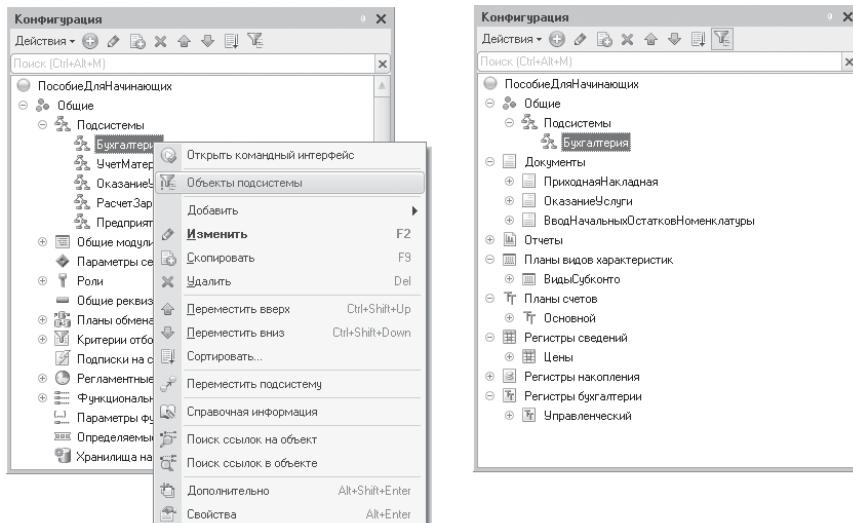


Рис. 14. Отбор объектов в дереве конфигурации по принадлежности к текущей подсистеме

Снять фильтр по подсистеме вы можете с помощью команды Отключить универсального диалога По подсистемам, вызываемого кнопкой По подсистемам  в командной панели окна дерева конфигурации.

Пример 10. Показать объекты из нескольких подсистем

Часто бывает, что вы хотите отобрать объекты по нескольким подсистемам сразу. Например, вам нужно показать в дереве конфигурации только те объекты, которые связаны с бухгалтерией и расчетом зарплаты.

В этом случае лучше всего использовать универсальный диалог По подсистемам.

Для этого нажмите кнопку По подсистемам в командной панели окна дерева конфигурации. В открывшемся окне Отбор по подсистемам снимите флажок с корня конфигурации, отметьте подсистемы Бухгалтерия и Расчет зарплаты и нажмите кнопку Установить. В результате будут показаны только те объекты конфигурации, которые относятся к этим подсистемам (рис. 15).

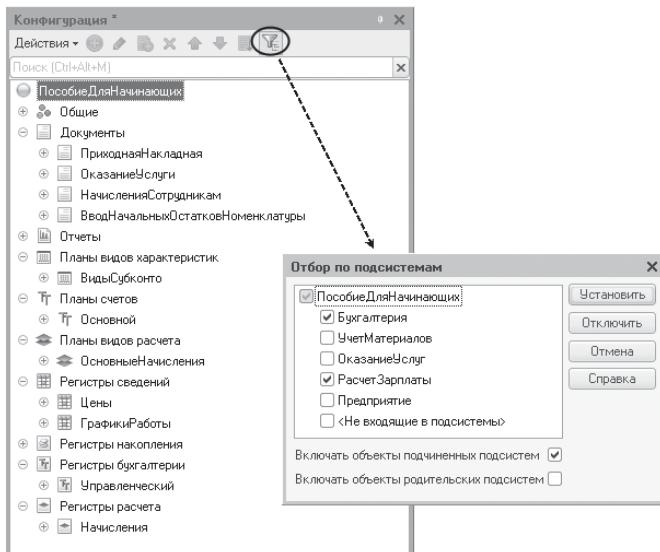


Рис. 15. Отбор объектов в дереве конфигурации по принадлежности к подсистемам

Снять фильтр по подсистемам вы можете кнопкой Отключить.

ПРИМЕЧАНИЕ

Также вы можете выполнить отбор объектов конфигурации по подсистемам с помощью команды Действия > По подсистемам в командной панели окна объектов конфигурации.

Пример 11. Палитра свойств в виде закладок

Стандартно свойства в палитре сортируются по категориям Основные, Использование, Оформление и т. д. (при этом кнопка Сортировка по категориям нажата). Группы свойств в палитре свойств выделены жирным шрифтом и располагаются списком, друг под другом.

Часто бывает, что вам нужно какое-то событие формы, но вы не помните точно его имя. Значит, вам нужна категория свойств События. Прокручивать палитру вниз не всегда удобно. Она может быть довольно большой, разных свойств может быть много.

В этом случае удобно представить палитру свойств в виде закладок. При этом каждая группа свойств будет размещаться на отдельной закладке. Чтобы переключить представление, на свободном месте палитры вызовите контекстное меню и выберите пункт Закладками (рис. 16).

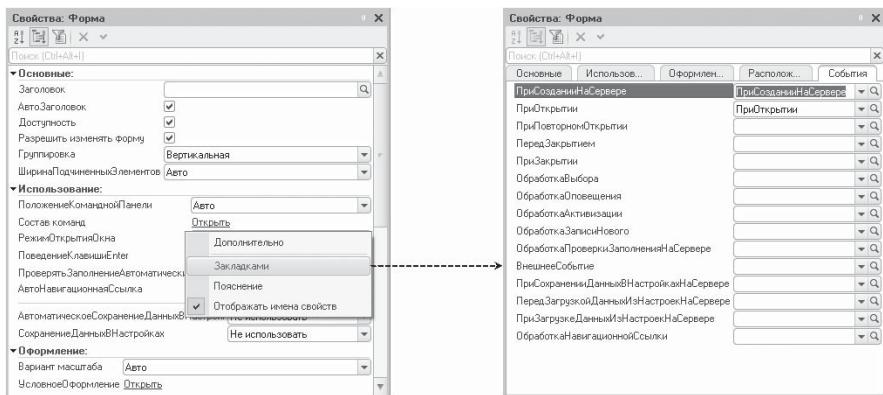


Рис. 16. Представление палитры свойств закладками

Теперь, быстро перейдя на закладку палитры свойств События, вы можете легко просмотреть все события формы, и вспомнить, какое из них вам нужно.

Для возврата к обычному виду палитры свойств вы можете выбрать пункт меню Списком.

ПРИМЕЧАНИЕ

Если свойства в палитре сгруппированы по категориям и представлены списком, то хорошим решением также может быть сворачивание ненужных групп свойств с помощью пиктограмм в виде маленькой стрелки, слева от названия группы. Оставив открытой только нужную группу свойств (События), гораздо проще и удобней визуально найти нужное вам свойство.

Пример 12. Показать синтакс-помощник для мобильного приложения

Особенность встроенного языка платформы «1С:Предприятия» такова, что далеко не все программные объекты доступны во всех контекстах исполнения. Например, многие объекты недоступны в управляемых формах или в мобильном приложении.

Недоступность объекта в конкретном контексте исполнения может выясниться только при синтаксической проверке модуля или при запуске конфигурации. Поэтому в конце описания объекта в синтакс-помощнике специально указывается, в каком контексте исполнения доступен данный объект.

Но заглядывать в описание и каждый раз прокручивать его до конца – долго и неудобно.

Гораздо эффективнее в данном случае настроить синтакс-помощник так, чтобы в нем отображались только те объекты, которые доступны в нужном вам контексте.

Предположим, вы разрабатываете мобильное приложение. В этом случае вам нужно ограничить контекст синтакс-помощника объектами, доступными только в мобильном приложении.

Для этого вы можете нажать кнопку Открыть режим настройки параметров , находящуюся в командной панели синтакс-помощника. И в окне настройки параметров синтакс-помощника оставить отметку только у режимов исполнения Мобильное приложение – клиент и Мобильное приложение – сервер (рис. 17).

В этом случае вам не придется каждый раз уточнять, доступен объект в мобильном приложении или нет. Если вы ищете в синтакс-помощнике какое-то свойство и не находитите, значит оно недоступно в мобильном приложении, и его нельзя использовать. Иначе вы получите ошибку при проверке модуля.

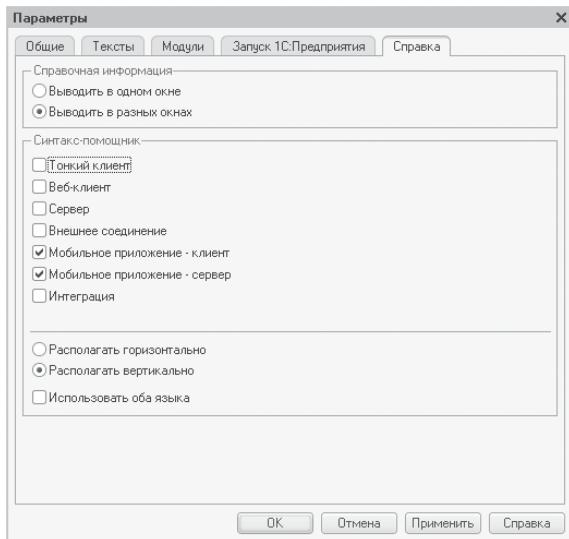


Рис. 17. Фильтрация объектов, показываемых в синтакс-помощнике

ПРИМЕЧАНИЕ

Возможна и другая ситуация. Например, вам ответили в конференции, что нужно использовать некоторый метод. Вы хотите почитать описание этого метода, но в синтакс-помощнике вы его не видите. Тогда вам нужно, наоборот, расширить контекст синтакс-помощника, потому что нужный контекст у вас, скорее всего, отключен.

Пример 13. Показать синтакс-помощник на двух языках

Все операторы встроенного языка имеют как русскоязычное, так и англоязычное написание, которое можно использовать одновременно в одном исходном тексте. Документация и синтакс-помощник содержат русскоязычный и англоязычный синтаксис и синонимы для всех конструкций встроенного языка.

Но стандартно в синтакс-помощнике отображается только русский язык. Правда, англоязычные названия свойств и методов отображаются в окне описания в скобках, рядом с русскими названиями, но их не видно в дереве на закладке Содержание, и по ним нельзя производить поиск.

Некоторые разработчики настолько привыкли к другим языкам программирования, что хотят видеть англоязычные наименования также и в дереве синтакс-помощника и выполнять по ним поиск.

В этом случае вы можете в настройках параметров синтакс-помощника (Сервис > Параметры > Справка) установить флажок Использовать оба языка (рис. 18).

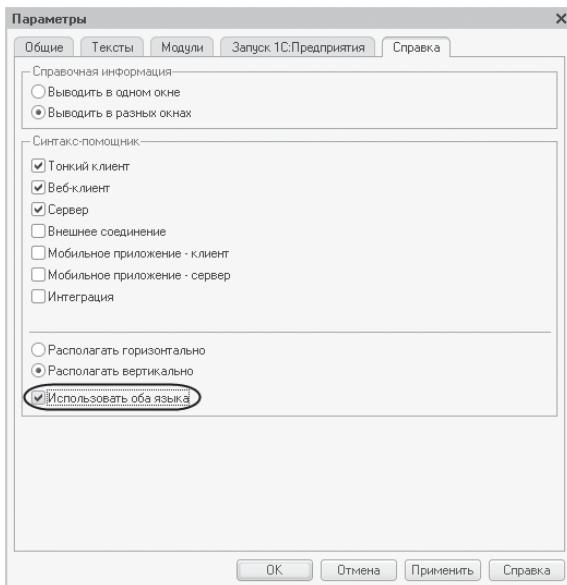


Рис. 18. Настройка параметров синтакс-помощника

После этого по англоязычным наименованиям так же, как и по русским, можно будет выполнять поиск. Результат будет одинаковым (рис. 19).

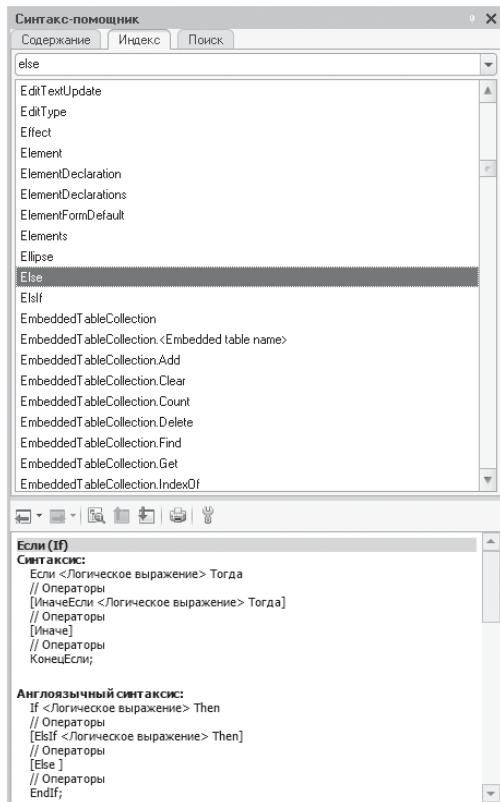


Рис. 19. Окно синтакс-помощника

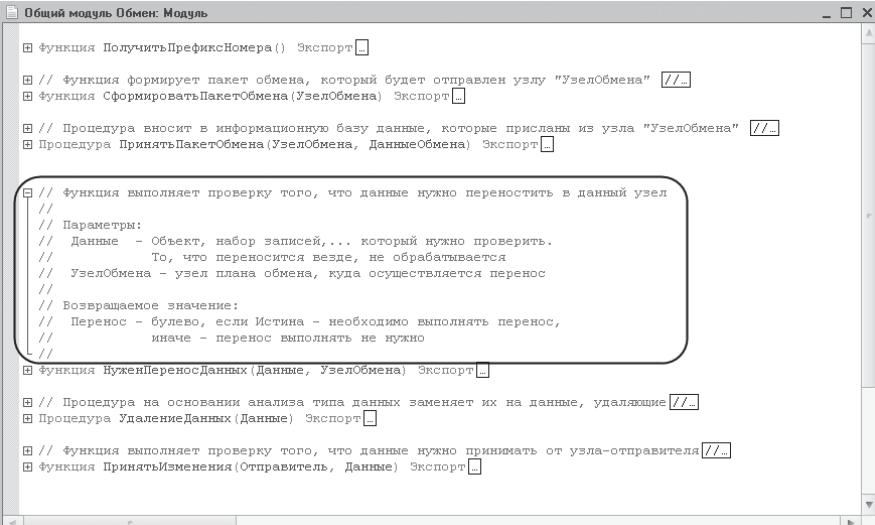
Пример 14. Скрыть содержимое циклов и условий

Предположим, вы редактируете текст длинного и сложного модуля. Чтобы визуально найти отлаживаемый фрагмент, приходится все время пролистывать текст модуля до нужного места. Это рассеивает внимание и мешает сосредоточиться.

Для облегчения этой задачи удобно использовать сворачивание синтаксических конструкций (процедуры и функции, комментарии к ним, циклы, условия и т. п.).

Чтобы свернуть или развернуть группу, достаточно щелкнуть мышью по маркеру группы. Если при этом нажать клавишу Ctrl, то будут также свернуты/развернуты и все вложенные в нее группы (циклы, условия и пр.).

Например, над заголовком процедуры в свернутом виде показывается группа описания процедуры, которую можно раскрыть и прочитать описание (рис. 20).



```
■ Общий модуль Обмен: Модуль
■ Функция ПолучитьПрефиксНомера() Экспорт ...
■ // Функция формирует пакет обмена, который будет отправлен узлу "УзелОбмена" //...
■ Функция СформироватьПакетОбмена(УзелОбмена) Экспорт ...
■ // Процедура вносит в информационную базу данные, которые присланы из узла "УзелОбмена" //...
■ Процедура ПринятьПакетОбмена(УзелОбмена, ДанныеОбмена) Экспорт ...

■ // Функция выполняет проверку того, что данные нужно перенести в данный узел
■ //
■ // Параметры:
■ //   Данные - Объект, набор записей,... который нужно проверить.
■ //   То, что переносится везде, не обрабатывается
■ //   УзелОбмена - узел плана обмена, куда осуществляется перенос
■ //
■ // Возвращаемое значение:
■ //   Перенос - булево, если Истина - необходимо выполнять перенос,
■ //           иначе - перенос выполнять не нужно
■ //

■ Функция НуженПереносДанных(Данные, УзелОбмена) Экспорт ...
■ // Процедура на основании анализа типа данных заменяет их на данные, удаляющиеся //...
■ Процедура УдалениеДанных(Данные) Экспорт ...
■ // Функция выполняет проверку того, что данные нужно принимать от узла-отправителя //...
■ Функция ПринятьИзменения(Отправитель, Данные) Экспорт ...
```

Рис. 20. Вид свернутых и развернутых групп в тексте модуля

Группировка синтаксических конструкций настраивается в параметрах конфигуратора: Сервис > Параметры > Модули > Группировка. Стандартно устанавливается группировка текста процедур и функций, их описания в виде комментариев, группировка областей модуля и их комментариев. При этом все эти группы, кроме текста областей, при открытии модуля показываются в свернутом виде (рис. 21).

Но если вам хочется большей лаконичности, можно включить в настройках параметров группировки модулей все флагшки и задать возможность группировки и сворачивания циклов и условий. После этого, открыв процедуру, мы увидим все циклы и условия в ней свернутыми (рис. 22).

Однако не всем это удобно. Поэтому, исходя из собственных предпочтений, вы можете настроить параметры группировки модулей (как и другие параметры конфигуратора) под себя и использовать их в дальнейшей работе.

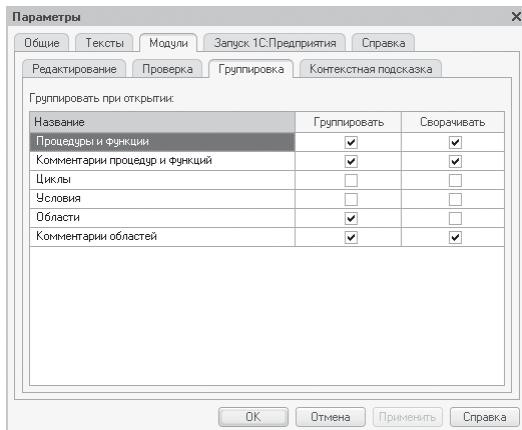


Рис. 21. Настройка группировки в тексте модуля

```

Функция ПолучитьПрефиксНомера() Экспорт
{
    // Функция формирует пакет обмена, который будет отправлен узлу "УзелОбмена" [/...]
    Функция СформироватьПакетОбмена(УзелОбмена) Экспорт
    {
        // Процедура вносит в информационную базу данные, которые присланы из узла "УзелОбмена" [/...]
        Процедура ПринятьПакетОбмена(УзелОбмена, ДанныеОбмена) Экспорт
        {
            ЧтениеXML = Новый ЧтениеXML;
            ЧтениеXML.УстановитьСтрока(ДанныеОбмена.Получить());
            ЧтениеСообщения = ПланОбмена.СоздатьЧтениеСообщения();
            ЧтениеСообщения.НачатьЧтение(ЧтениеXML);
            ПланыОбмена.УдалитьРегистрацияИзменений(ЧтениеСообщения.Отправитель, ЧтениеСообщения.НомерПринятого);

            НачатьТранзакцию();
            Пока ВозможностьЧтенияXML(ЧтениеXML) Цикл
            {
                ЗаписьСообщения.ЗакончитьЧтение();
                ЧтениеXML.Закрыть();
            }
            КонецПроцедуры
        }
        // Функция выполняет проверку того, что данные...
        Функция НуженПереносДанных(Данные, УзелОбмена) Экспорт
    }
}

```

Рис. 22. Вид свернутых и развернутых групп в тексте модуля

Установив курсор на свернутую группу, удобно, даже не выделяя ее, скопировать и вставить группу целиком через буфер обмена (**Ctrl + C/Ctrl + V**).

ПРИМЕЧАНИЕ

Чтобы свернуть сразу все группы в тексте модуля, вы можете использовать кнопку **Свернуть все группы** в панели инструментов конфигуратора.

Пример 15. Объединить несколько процедур в группу

Для облегчения понимания (для себя или других разработчиков) часто бывает нужно выделять в тексте логические блоки, которые не связаны с используемыми синтаксическими конструкциями. Например, внутри одной процедуры хочется выделить часть операторов, выполняющих законченную функцию, самостоятельную часть алгоритма. Или, наоборот, хочется как-то показать, что группа процедур/функций относится к определенной функциональности или к определенному алгоритму.

Можно пытаться как-то выделять это комментариями... Но такой способ неудобен.

Гораздо лучше для таких задач использовать именованные области текста модуля. По сравнению с комментариями явное преимущество областей в том, что их можно сворачивать, к тому же области могут быть вложены друг в друга, что делает текст модуля более структурированным и легким для понимания.

Для выделения областей используются инструкции препроцессора #Область <Имя области>... #КонецОбласти. Имя области вы задаете сами, оно может быть произвольным, но должно удовлетворять правилам встроенного языка. Желательно, чтобы имя области отражало назначение области. Следует учитывать, что области не должны пересекаться с другими областями и синтаксическими конструкциями.

Например, вам нужно разбить на логические разделы текст модуля формы документа. Для того чтобы модуль был более понятен и проще воспринимался бы другими разработчиками, которым, возможно, понадобится вносить в него изменения при последующей доработке прикладного решения.

Процедуры модуля обычно принято группировать по функциональному признаку. Например, в отдельные группы обычно объединяются процедуры-обработчики событий, обработчики команд, служебные процедуры и функции модуля и т. п.

В соответствии с этим вы можете создать, например, в модуле формы документа три области: ОбработчикиСобытий (1), ОбработчикиКомандФормы (2) и СлужебныеПроцедурыИФункции (3). Внутри области ОбработчикиСобытий поместите две вложенные области: ОбработчикиСобытийФормы (1-а – на рисунке она свернута) и ОбработчикиСобытийЭлементовФормы (1-б – развернута), которые не пересекаются друг с другом (рис. 23).

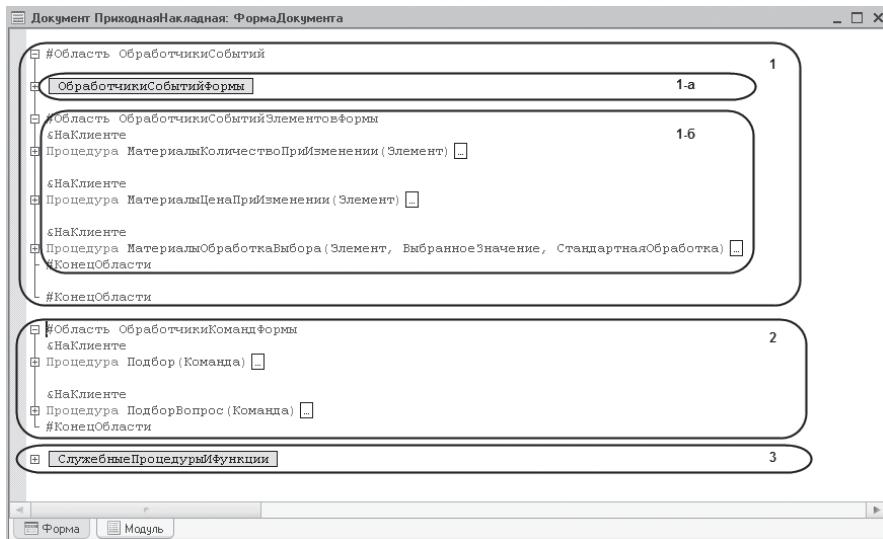


Рис. 23. Модуль формы с выделенными разделами в виде областей

В результате модуль стал более организован и прост в сопровождении не только для других разработчиков, но и для вас. Потому что теперь, открыв модуль, можно развернуть только отдельные нужные вам группы и увидеть содержимое в более логичном и компактном виде.

Иногда бывает нужно, наоборот, выделить фрагменты алгоритма в рамках одной процедуры или функции. Например, длинную процедуру обработки проведения документа хочется разбить на этапы, выполняющие разные функциональные задачи. Для описания назначения программных областей перед началом области (#Область) вы можете поместить комментарий, который также будет группироваться (рис. 24).

Таким образом, при открытии текста процедуры вы можете развернуть только нужную вам область процедуры и работать с ней. О назначении остальных областей вы сможете судить по комментариям к ним, если это непонятно из имени области, как это было в случае с выделением разделов модуля (см. рис. 23 – здесь комментарии областей избыточны).

В результате текст процедуры можно будет «охватить одним взглядом». Это, безусловно, облегчит процесс восприятия и понимания текста процедуры, поиск в ней ошибок и т. д.

```
Документ ОказаниеУслуги: Модуль объекта
МенеджерВТ = Новый МенеджерВременныхТаблиц;
// 1-ый этап.
// Формирование временной таблицы, содержащей перечень номенклатуры документа.
#Область НоменклатураДокумента
Запрос = Новый Запрос;

Запрос.Текст =
"ВЫБРАТЬ
|   ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
|   ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК ВидНоменклатуры,
|   ОказаниеУслугиПереченьНоменклатуры.НаборСвойств,
|   СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК КоличествоВДокументе,
|   СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК СуммаВДокументе
| ПОМЕСТИТЬ НоменклатураДокумента
| ИЗ
|   Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК ОказаниеУслугиПереченьНоменклатуры
| ГДЕ
|   ОказаниеУслугиПереченьНоменклатуры.Ссылка = «Ссылка
|
| СГРУППИРОВАТЬ ПО
|   ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
|   ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры,
|   ОказаниеУслугиПереченьНоменклатуры.НаборСвойств";
Запрос.УстановитьПараметр("Ссылка", Ссылка);

РезультатЗапроса = Запрос.Выполнить();
#КонецОбласти

// 2-ой этап.
// Расчет стоимости номенклатуры и формирование движений в регистрах накопления.
# ДвиженияДокумента

// 3-ий этап.
//Контроль остатков номенклатуры при оперативном проведении документа.
# КонтрольОстатков
КонецПроцедуры
```

Рис. 24. Процедура в модуле объекта, логически разделенная на три этапа проведения документа

Получение помощи и подсказок

Все держать в голове – просто невозможно, да и не нужно. Поэтому для вас важно умение пользоваться теми инструментами для получения помощи и подсказок, которые содержит платформа «1С:Предприятие». В каждой конкретной ситуации наиболее эффективен тот или иной вид помощи.

Прежде всего, для быстроты и правильности написания кода полезно пользоваться контекстной подсказкой, которая появляется при наборе текста программы. Это позволит вам избежать многих досадных ошибок, таких как пропущенная буква в имени переменной и т. п.

Кроме того, для помощи разработчику в конфигураторе существует синтаксис-помощник, из которого вы можете получить всю необходимую информацию. Например, вы хотите в модуле использовать какой-то метод, но не помните, как он вызывается. Тогда вы можете сразу же из модуля перейти в синтаксис-помощник к описанию этого метода.

Если вы хотите в палитре свойств установить какое-то свойство, то, чтобы прочитать информацию о нем, вы можете сразу же из палитры свойств перейти к описанию свойства в синтаксис-помощнике. Кроме того, в синтаксис-помощнике вы можете выполнить произвольный поиск по фрагментам слов в описании. Если этого недостаточно, можно также перейти из синтаксис-помощника к методической информации в Интернете, на сайте ИТС.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 16. Подсказки при наборе текста программы

Если вы знаете хотя бы начало имени реквизита, переменной, параметра и т.п. и этот тип объекта точно известен в контексте редактируемого модуля, то в этом случае наиболее эффективно, не отрываясь от клавиатуры, набрать несколько первых символов имени, свойства или метода нужного программного объекта и воспользоваться контекстной подсказкой.

Вы можете вызвать ее с помощью комбинации клавиш **Ctrl + <Пробел>** на любом этапе набора текста. Список контекстной подсказки включает имена системных перечислений, предопределенных элементов, свойств и методов программных объектов, наименования объектов, определенных в конфигурации, и пр. В начале каждой строки списка присутствуют пиктограммы, указывающие на вид объекта: например, черная черта – это свойства глобального контекста, красная черта – локальные переменные модуля, зеленые символы Р() – процедуры прикладных объектов и т. д. (рис. 25).

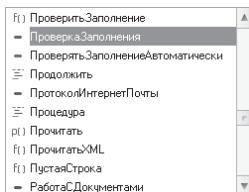


Рис. 25. Список контекстной подсказки

При вызове контекстной подсказки список будет позиционирован на первом найденном соответствии с набранным текстом или ближайшем к нему варианте. Список можно пролистать вручную, выделить и выбрать из него нужный вариант щелчком мыши или нажатием клавиши **Enter**. Если продолжить набор при открытом списке контекстной подсказки, то по мере набора текста он позиционируется на следующем соответствии и т. д.

Пример 17. Почему контекстная подсказка подсказывает не всегда

Заметьте, что контекстная подсказка работает не всегда, а только тогда, когда тип объекта точно известен в контексте редактируемого модуля.

Если объект конструируется в этом же модуле, то контекстная подсказка для него всегда работает. Например, при создании нового табличного документа в модуле формы обработки и дальнейшем обращении к нему средствами

встроенного языка в списке контекстной подсказки вам доступны все свойства и методы этого табличного документа (рис. 26).

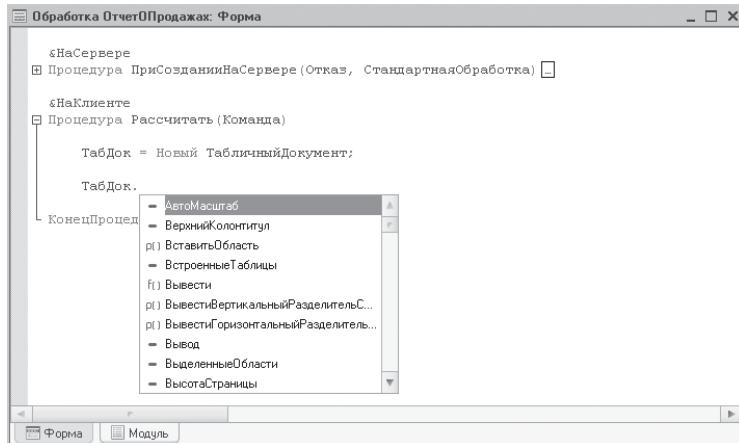


Рис. 26. Список контекстной подсказки для табличного документа, созданного в модуле

Если же вы вызовете процедуру общего модуля и передадите туда табличный документ в качестве параметра, то в общем модуле контекст табличного документа будет пустым (рис. 27).

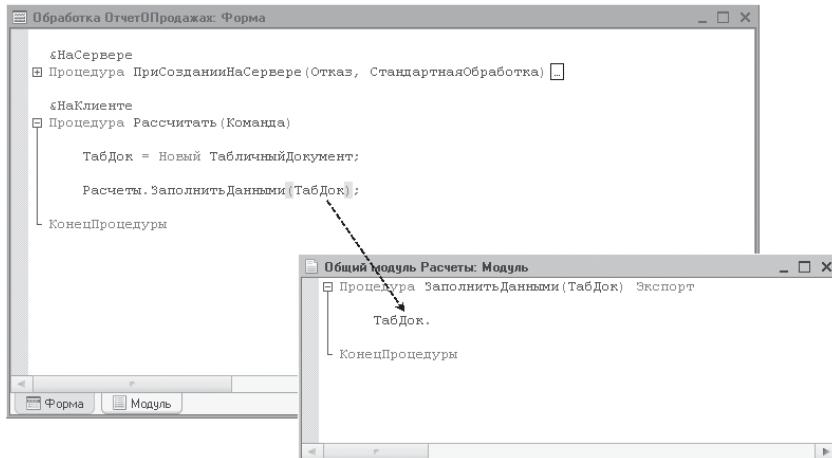


Рис. 27. Отсутствие контекстной подсказки для табличного документа, переданного в процедуру общего модуля в качестве параметра

Пример 18. Быстро открыть описание метода в синтакс-помощнике

Предположим, что вы дорабатываете какую-то незнакомую конфигурацию и вам нужно модифицировать процедуру, из которой вызывается метод объекта, имеющий несколько параметров. Например, метод ВыбратьИзменения() менеджера планов обмена. И, предположим, вы не знаете, какие параметры надо передавать в этот метод.

Для этого необязательно открывать окно синтакс-помощника и искать метод на закладке Индекс или в дереве синтакс-помощника. Все эти действия можно выполнить одним нажатием, используя возможность контекстной помощи, которая предоставляется редактором текста модулей. То есть из текста модуля вы можете быстро перейти в синтакс-помощник к описанию текущей синтаксической конструкции, находящейся «под курсором». Это очень удобно и наиболее эффективно для решения поставленной задачи.

Например, установите курсор на методе ВыбратьИзменения() менеджера планов обмена и нажмите кнопку Поиск строки в Синтакс-помощнике (Ctrl + F1) в панели инструментов конфигуратора. Будет произведен поиск этого идентификатора в синтакс-помощнике среди конструкций встроенного языка (рис. 28).

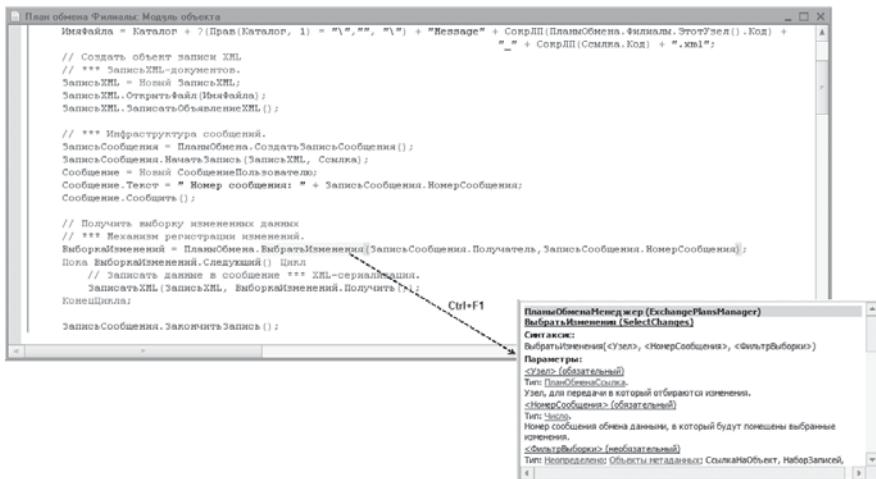


Рис. 28. Использование контекстной помощи синтакс-помощника

Если при поиске будет найдено несколько страниц с описанием искомого идентификатора, то будет открыто окно со списком страниц. В нем

можно найти нужную страницу визуально или с помощью поиска (Ctrl + F). Подробнее об этом рассказывается в примере «Найти свойство или метод в синтакс-помощнике» на стр. 66.

Таким образом, вызывая контекстную помощь синтакс-помощника из непонятных вам мест модуля, можно разобраться в коде, написанном другим разработчиком.

ПРИМЕЧАНИЕ

Следует иметь в виду, что назначаемые обработчики элементов формы содержат стандартные имена, включающие еще и префиксы элементов, к которым они относятся. Например: СписокФайловПриАктивизацииСтроки, ГруппаСтраницыПриСменеСтраницы. Поэтому если встать на такой идентификатор и нажать Ctrl + F1, в синтакс-помощнике ничего найдено не будет.

Чтобы найти описание стандартного обработчика, в данном случае нужно выделить ту часть идентификатора, которая содержит его имя (например, ПриАктивизацииСтроки, ПриСменеСтраницы) и после этого нажать Ctrl + F1.

Пример 19. Найти открытое описание в дереве синтакс-помощника

Предположим, в форме некоторой обработки вам нужно создать обработчик события, возникающего при перетаскивании в поле табличного документа. В палитре свойств этого поля вы видите список событий, но не уверены, какое из них вам нужно.

В этом случае лучше сначала выделить одно из событий (например, Перетаскивание) и перейти из палитры свойств в синтакс-помощник (см. пример «Быстро найти в синтакс-помощнике описание свойства из палитры свойств» на стр. 42, чтобы прочитать информацию об этом событии).

Если из описания выяснится, что это не то событие, которое вам нужно, вы можете быстро открыть и увидеть описание других событий поля табличного документа. Для этого проще всего нажать кнопку Найти текущий элемент в дереве , находящуюся в командной панели синтакс-помощника (рис. 29).

При этом на закладке Содержание будет раскрыта ветвь дерева синтакс-помощника, соответствующая текущему описанию. В нашем примере это группа событий расширения табличного документа для поля управляемой формы. Таким образом вы можете быстро увидеть и прочитать информацию о других событиях этого же объекта.

ПРИМЕЧАНИЕ

Возможность найти текущий элемент в дереве очень полезна также для того, чтобы лучше ориентироваться в структуре программных объектов,

когда вы знаете имя свойства или метода, но не знаете, какому объекту они принадлежат. Тогда вы находитите нужное свойство по индексу синтакс-помощника и переходите в дерево с помощью соответствующей кнопки.

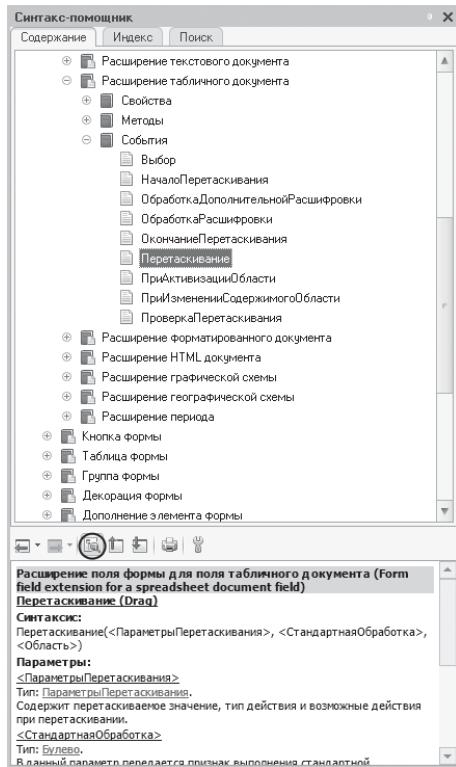


Рис. 29. Переход к текущему описанию в дереве синтакс-помощника

Пример 20. Быстро найти в синтакс-помощнике описание свойства из палитры свойств

Предположим, в интернет-конференции вам сказали, что надо установить свойство ФорматРедактирования для поля ввода в форме. Вы нашли это свойство в палитре свойств поля формы, но не знаете, как его задавать.

В этом случае за дополнительной информацией можно обратиться в синтакс-помощник. Но самостоятельно открывать синтакс-помощник и искать там описание свойства неудобно.

Легче и быстрее перейти в синтакс-помощник непосредственно из палитры свойств. Для этого выделите искомое свойство (например, ФорматРедактирования), вызовите контекстное меню и выберите пункт Справка (рис. 30).

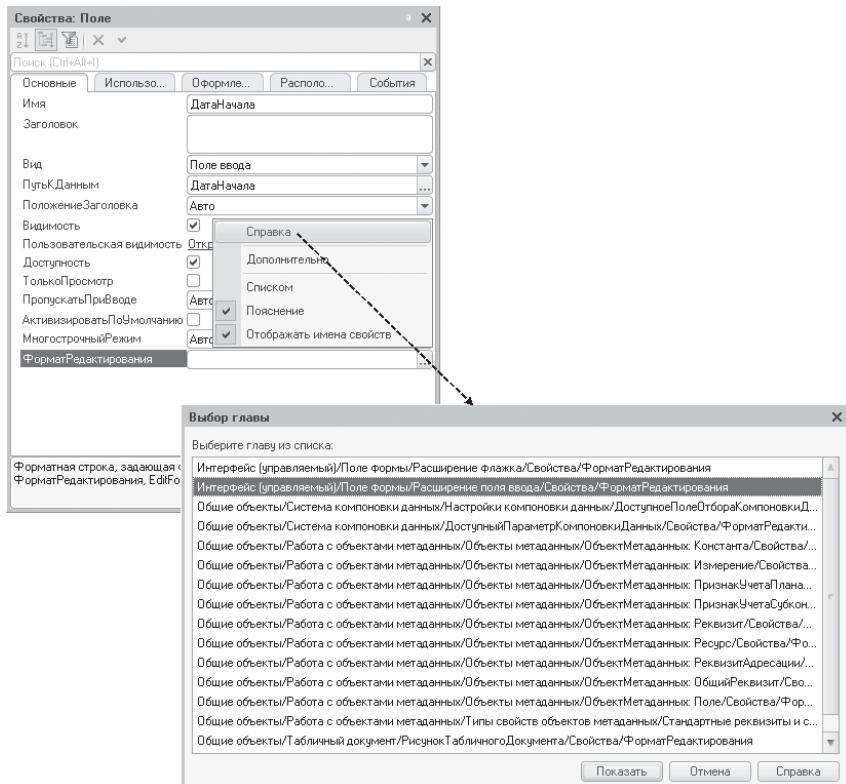


Рис. 30. Вызов синтакс-помощника из палитры свойств

Если в синтакс-помощнике содержится описание выбранного свойства сразу для нескольких объектов, то будет открыто окно со списком страниц синтакс-помощника, в которых оно описывается. В этом окне вы можете найти нужную страницу визуально или с помощью поиска ($Ctrl + F$) и открыть описание нужного свойства в синтакс-помощнике. Подробнее об этом рассказывается в примере «Найти свойство или метод в синтакс-помощнике» на стр. 66.

Нужно сказать, что данная возможность (пункт Справка в контекстном меню) есть не у всех свойств, а только у тех, которые доступны

во встроенным языке. Некоторые свойства объектов конфигурации доступны только для визуального конструирования, поэтому они не описаны в синтакс-помощнике.

Пример 21. Что делать, когда описания в синтакс-помощнике недостаточно

Предположим, вам нужно связать некоторый элемент данных с предопределенным элементом справочника. И предположим, в интернет-конференции вам сказали, что для этого нужно использовать свойство ИмяПредопределенныхДанных объекта СправочникОбъект.

Вы прочитали информацию об этом свойстве в синтакс-помощнике, но все равно не поняли, как его использовать. Нужен пример использования свойства ИмяПредопределенныхДанных во встроенном языке.

В этом случае вам может помочь ссылка Методическая информация, которая находится в конце каждого описания (рис. 31).

По этой ссылке открывается окно браузера, в котором подобраны методические материалы для выбранного раздела. Источниками материалов являются: ИТС, партнерская конференция, база знаний по технологическим вопросам крупных внедрений, сайт «1С:Предприятие», конференция начинающих разработчиков и др. Ссылки на методические материалы постоянно обновляются (рис. 32).

В нашем примере вверху окна содержатся ссылки на документацию, а под ними – ссылка на полезную статью на ИТС об особенностях предопределенных данных.

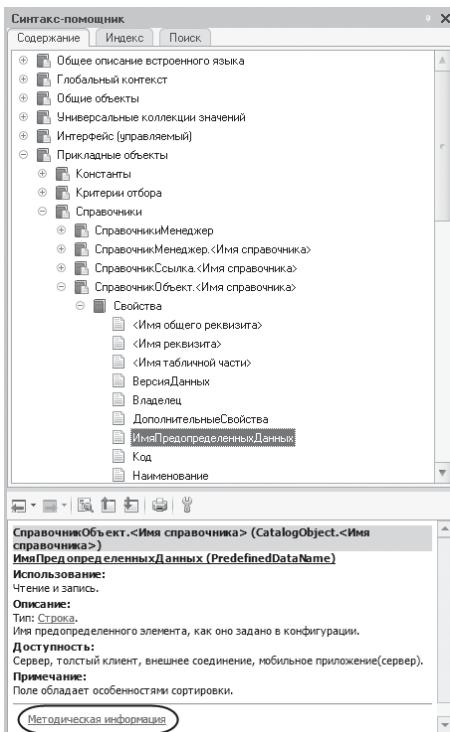


Рис. 31. Переход к методической информации из синтакс-помощника

The screenshot shows a search results page for the query 'Как отказаться от использования модальности?'. The main content area displays the title 'ИмяПредопределенныхДанных' and a detailed description of the feature, mentioning its availability in V8Update 8.3.4.365 and V8Update 8.3.3.721, and its compatibility with the 'Compatibility mode'.

Поиск по категориям

Как отказаться от использования модальности?

Прикладные объекты/Справочники/СправочникОбъект.<Имя справочника>/Свойства/ИмяПредопределенныхДанных

5 [новое] Документация 8.3: 5.6.5.1. Предопределенные данные. Общая информация

5 [новое] Документация 8.3: 5.6.5.2. Предопределенные данные. Работа в распределенной информационной базе

V8Update 8.3.4.365: Прикладные объекты. Для объектов конфигурации, которые могут содержать предопределенные данные, реализована возможность устанавливать.

V8Update 8.3.3.721: Прикладные объекты. Общие реквизиты / Разделение данных. Права доступа. Обмен данными. Мобильное приложение. Реализована возможность.

ИТС: Особенности предопределенных элементов объектов метаданных при работе с отключенным режимом совместимости (Раздел обновлен)

<http://its.1c.ru/db/METHOD8DEV#content:5367:1>

Рис. 32. Подборка методических материалов на интернет-ресурсах фирмы «1С»

Пример 22. Произвольный поиск в синтакс-помощнике

Иногда может понадобиться найти что-то в синтакс-помощнике не по названиям свойств и методов, а по каким-то словам, которые есть только внутри описаний. Для этого предназначена закладка Поиск.

Например, вы знаете, что где-то можно включить механизм разделения итогов, но не помните, где именно. Вы попытались с помощью поиска по индексу найти «разделениеитогов», но поиск не принес вам желаемых результатов (рис. 33).

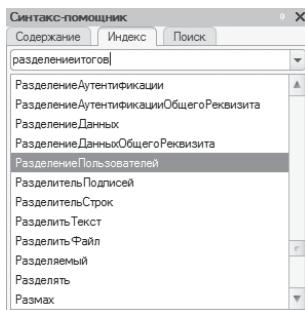


Рис. 33. Попытка поиска по индексу в синтакс-помощнике

Тогда вы можете на закладке Поиск синтакс-помощника набрать «разделение итогов», и в список результатов поиска попадут те объекты и методы, в описании которых встречается искомая фраза. В описании найденные фрагменты будут выделены цветом (рис. 34).

Поиск в синтакс-помощнике другими способами в аналогичных ситуациях может занять у вас слишком времени и сил или вообще не дать нужных результатов.

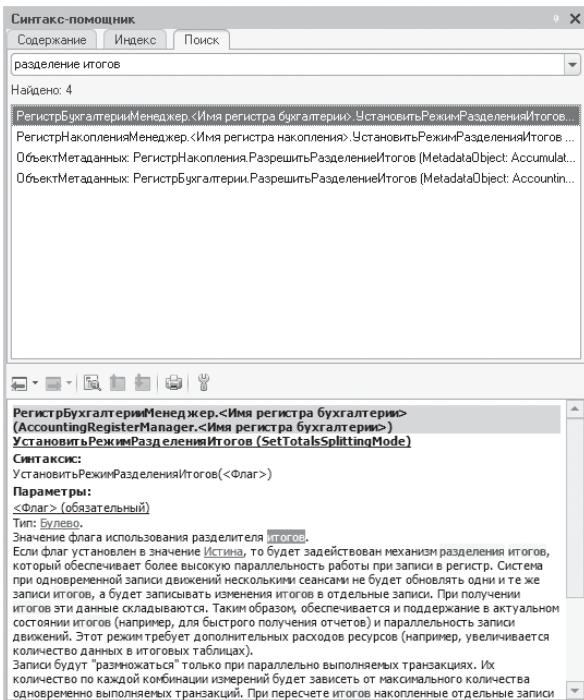


Рис. 34. Поиск по синтаксис-помощнику на закладке «Поиск»

Закладка Поиск также может быть полезна в случае, когда нужно быстро найти объекты, методы, свойства, относящиеся к одной теме. Например, вы помните, что при работе с текстовыми файлами где-то, в каких-то методах можно было задать кодировку текста. Но что это были за объекты и что это были за методы, вы не помните. Зато помните, что речь шла о кодировке UTF-8. В этом случае вы можете набрать «UTF-8» на закладке Поиск и сразу найти те объекты и методы, которые вам нужны (рис. 35):

- ЗаписьТекста.Открыть(),
- ЧтениеТекста.Открыть(),
- ТекстовыйДокумент.Прочитать(),
- ТекстовыйДокумент.Записать().



Рис. 35. Поиск по синтакс-помощнику на закладке «Поиск»

Искать это другими способами – очень долго или вообще невозможно.

Найти что-то

Прежде чем что-то изменить, вам нужно сначала найти это «что-то» в конфигурации. Например, вы хотите изменить какое-то свойство какого-то объекта. Чтобы это сделать, нужно сначала найти этот объект в дереве объектов конфигурации, открыть его палитру свойств, найти в ней нужное свойство и затем установить значение этого свойства.

От того, насколько просто и быстро будет выполнен поиск, напрямую зависит эффективность разработки вообще. Потому что иногда в процессе поиска можно забыть, что, собственно, хотелось сделать...

Для поиска объекта в дереве конфигурации или свойства в палитре свойств существует множество способов. В зависимости от ситуации удобнее применять тот или иной способ поиска. Вы можете искать при помощи строки поиска в дереве конфигурации или в палитре свойств. Можете искать визуально, по алфавиту, можете воспользоваться глобальным поиском по конфигурации и т. п.

Кроме того, в данном разделе мы рассмотрим, как быстро найти нужную информацию в синтакс-помощнике, и как наиболее удобно искать что-то в текстах модулей.

При исправлении ошибок, которые неизбежно встречаются как при проверке модулей, так и при исполнении конфигурации, важно быстро найти то место, где обнаружена ошибка. В большинстве случаев вы можете перейти к нему автоматически из сообщения об ошибке.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 23. Как понять что искать

Предположим, вам нужно что-то изменить в поведении прикладного решения или просто понять, почему прикладное решение ведет себя так, а не иначе. Для этого вам нужно найти объект конфигурации, который за это «отвечает», и посмотреть или изменить соответствующим образом его свойства, модули, формы и т. п.

Перед тем как выполнять поиск, нужно определиться, по какому свойству объекта – имени, синониму или представлению – его искать.

Например, вы работали с прикладным решением и заметили, что форма с заголовком «Накладная» ведет себя как-то не так. Или вы спросили об этом в конференции, а вам ответили, что нужно смотреть, что делает «Накладная».

Казалось бы, все просто. Нужно открыть конфигуратор и найти там объект, который называется «Накладная». Но на самом деле это не совсем так.

В конфигураторе, в дереве объектов конфигурации, каждый объект имеет свое уникальное имя, которое ему дает разработчик. Эти имена предназначены для того, чтобы использовать их в процессе конфигурирования прикладного решения, чтобы обращаться к объектам конфигурации в программных алгоритмах.

Как правило, обычный прикладной пользователь, например бухгалтер, видит в интерфейсе прикладного решения не имена, а синонимы или представления (представления объекта, представления списка и др.) объектов конфигурации.

Имя объекта конфигурации задается разработчиком при создании объекта конфигурации. На основании имени платформа автоматически заполняет его синоним. Теоретически разработчик может, конечно, совсем удалить синоним, который создала платформа, но обычно так не делают, а редактируют синоним, чтобы он был понятен пользователю. Скорее всего, именно его увидит пользователь в интерфейсе.

Бывают ситуации, когда одного синонима недостаточно для того, чтобы удобно представить объект конфигурации в интерфейсе. Например, объект конфигурации имеет синоним «Накладная». Но хотелось бы, чтобы команда перехода к списку накладных называлась «Накладные», а форма, открывающаяся по этой команде, имела бы заголовок «Реестр накладных». Для этого разработчик использует представления объекта – устанавливает свойства Представление списка как Накладные и Расширенное представления списка как Реестр накладных.

Поэтому, когда вам нужно найти в конфигураторе некую «накладную», прежде всего вы должны попытаться понять, что такое «Накладная»? Это имя, синоним или одно из представлений объекта конфигурации? От этого зависит способ, которым мы будем искать объект в конфигураторе.

Есть пара признаков, по которым можно с большой долей вероятности сделать правильный вывод.

Во-первых, если слово «Накладная» вы увидели в интерфейсе, то практически со 100 % вероятностью это либо синоним, либо одно из представлений объекта конфигурации. Чисто теоретически имя объекта конфигурации может «просочиться» в интерфейс. Если не указан ни синоним, ни одно из представлений. Но такая ситуация очень маловероятна. На практике синонимы есть у всех объектов конфигурации.

Во-вторых, это количество слов и способ их написания. Если, например, в конференции вам сказали, что нужно искать «РасходнаяНакладная», то это имя объекта. А если «Расходная накладная», то это либо синоним, либо одно из представлений. Потому что в соответствии со стандартами формирования имен объектов конфигурации в «1С:Предприятии» имя объекта не должно содержать пробелов, знаков препинания и других специальных символов. Если имя состоит из нескольких слов, то пробелы между ними удаляются, а первые буквы слов делаются прописными.

Пример 24. Самый простой способ поиска объектов

Предположим, в дереве объектов конфигурации вам нужно найти объект «Клиент». Поскольку это одно слово, то «Клиент» может быть как именем, так и синонимом или представлением объекта. И допустим, что вы не знаете точно, что это за объект – справочник, отчет, обработка и т. п.

Первое, что приходит в голову – открыть дерево объектов конфигурации и, просматривая ветви за ветвью, искать нужный объект визуально. Но в дереве конфигурации отображаются только имена объектов. А вы не знаете точно, что вы ищете и в какой ветви дерева это находится.

Поэтому наиболее эффективно в данном случае воспользоваться строкой поиска в дереве объектов конфигурации. При этом в дереве будут отобраны те объекты, которые либо в имени, либо в синониме, либо в комментарии содержат исковую строку.

Откройте дерево объектов конфигурации, установите курсор в строку поиска вверху окна и введите исковое слово «Клиент». По мере набора символов поиск будет выполняться автоматически. Результаты поиска будут показаны

тут же, в дереве конфигурации, причем найденные фрагменты слов в именах объектов будут подсвечены (рис. 36).

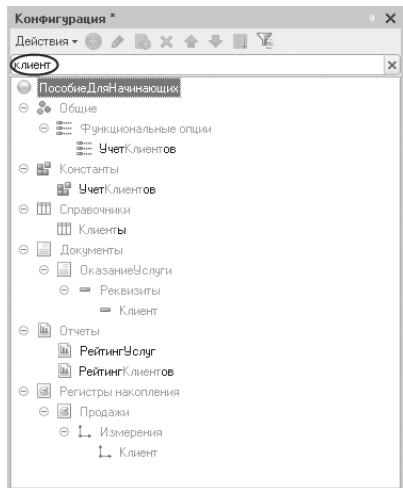


Рис. 36. Поиск объектов в строке поиска окна дерева конфигурации

ПРИМЕЧАНИЕ

Для демонстрации большинства примеров для большей наглядности используется реально работающая конфигурация, разработанная для книги «Практическое пособие разработчика».

Подсветка работает следующим образом:

- найденные фрагменты слов в имени объекта конфигурации подсвечиваются красным цветом непосредственно в дереве конфигурации;
- если искомые фрагменты найдены в синониме или в комментарии, то имя объекта в дереве конфигурации отображается черным цветом (в нашем примере искомая строка содержится в синониме отчета РейтингУслуг);
- если искомые слова найдены только в подчиненном объекте конфигурации, то имя родительского объекта отображается серым цветом.

Чтобы снять фильтр по найденным объектам конфигурации, вы можете нажать кнопку очистки (крестик) в строке поиска.

Строчку поиска можно ввести не полностью или ввести фрагменты слов, разделенные пробелом. При этом чем точнее указана строка поиска, тем меньше список найденных объектов и тем проще в нем ориентироваться.

Таким образом, когда непонятно, что вы ищете (имя, синоним или представление) и тип объекта заранее неизвестен, то показанный способ поиска является наиболее универсальным и удобным.

ПРИМЕЧАНИЕ

Чтобы быстро перейти в строку поиска в дереве конфигурации, вы можете нажать Ctrl + Alt + M.

Пример 25. Найти объект по имени в небольшой конфигурации

Предположим, вам нужно найти документ с именем «ПриходнаяНакладная». То есть вы точно знаете имя объекта и знаете, в какой ветви дерева его искать (в нашем примере Документы).

Если объектов в конфигурации немного, проще всего просто раскрыть ветвь Документы в дереве объектов конфигурации и визуально найти среди них нужный элемент.

Однако если конфигурация содержит много объектов, то искать объект «глазами» довольно утомительно. Это может отнять много времени и внимания.

Чтобы не пропустить случайно нужный элемент, удобно искать объект по первой букве имени (или по первым двум буквам имени, если набирать их быстро друг за другом) среди этого вида объектов.

Для этого раскройте ветвь Документы в дереве объектов конфигурации и нажмите букву «П». При каждом следующем нажатии этой буквы на клавиатуре курсор будет последовательно переходить к документам, имя которых начинается с буквы «П».

Также для облегчения поиска можно отсортировать по алфавиту список документов конфигурации, чтобы уже в отсортированном списке быстрее и проще найти то, что нужно.

Но прежде чем это делать, нужно учитывать некоторые моменты, о которых рассказывается в разделе «Найти объект по алфавиту».

Для сортировки объектов в конкретной ветви дерева конфигурации вы можете выделить эту ветвь и нажать кнопку Упорядочить список  в командной панели окна дерева объектов конфигурации (рис. 37).

После этого вы можете пролистать список объектов до нужной буквы или нажать первую букву имени и затем визуально найти нужный элемент.

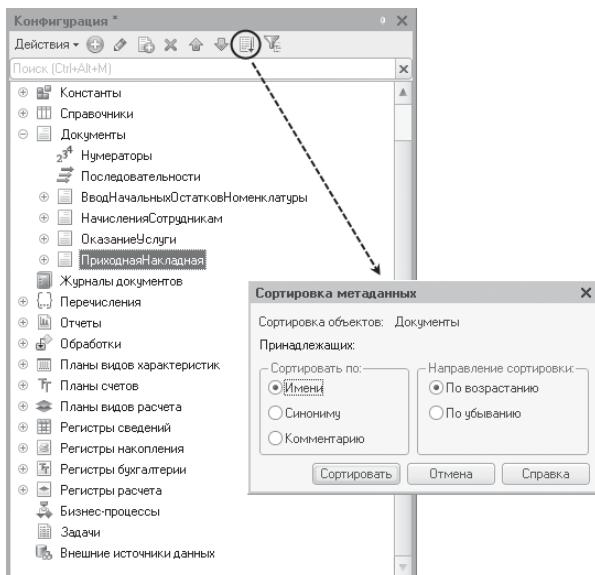


Рис. 37. Сортировка объектов в дереве объектов конфигурации

ПРИМЕЧАНИЕ

Когда вы перешли на клавиатуру, то для других действий с деревом также удобнее использовать клавиатуру, а не мышь. Перемещаться по дереву можно с помощью стрелок вверх/вниз, а чтобы раскрыть или, наоборот, свернуть ветку дерева можно использовать стрелки вправо/влево.

О назначении «горячих» клавиш в конфигураторе можно почитать во встроенной справке: Справка > Содержание справки > 1С:Предприятие > Сочетание клавиш (Конфигуратор).

Пример 26. Найти объект по синониму или представлению в небольшой конфигурации

Предположим, в интерфейсе прикладного решения, в разделе справочной информации, вы открыли форму элемента справочника «Графики работы сотрудников», в поведении которой вы хотите что-то изменить. Для этого в дереве объектов конфигурации вам нужно найти справочник с таким синонимом. То, что это синоним, а не имя, понятно из пробелов между словами.

Название «Графики работы сотрудников» может в общем-то быть и представлением. Но в форме элемента справочника, как правило, отображается представление объекта, которое для справочников обычно задается в единственном числе («График работы сотрудников»).

То есть мы ищем определенный объект (в нашем примере Справочник) по его синониму или представлению.

Синоним объекта, в отличие от его имени, не видно в дереве объектов конфигурации. Но, как вы знаете, все свойства объекта конфигурации можно увидеть в палитре его свойств.

Поэтому для решения поставленной задачи наиболее эффективно использовать палитру свойств объектов, расположенных на определенной ветви дерева.

Для этого раскройте ветвь дерева конфигурации Справочники, выделите первый справочник в списке, откройте его палитру свойств и выделите в ней свойство Синоним (рис. 38).

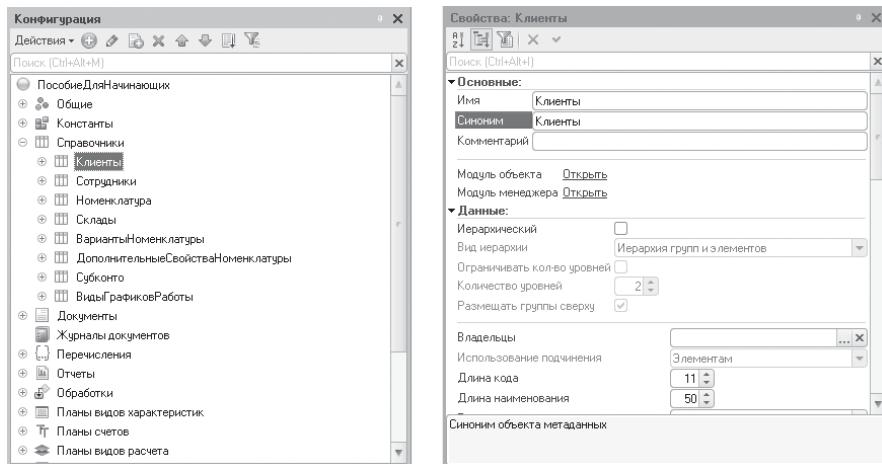


Рис. 38. Палитра свойств текущего объекта конфигурации

Затем вернитесь в дерево объектов конфигурации и стрелкой вниз «пробежитесь» по всем объектам открытой ветви дерева.

Свойство, которое вы выделили в палитре, останется «закрепленным» и всегда будет находиться в «зоне видимости». А его значение будет соответствовать тому объекту конфигурации, на котором вы находитесь в данный

момент. Таким образом, пробегая по дереву, в палитре свойств вы последовательно увидите синонимы объектов конфигурации (рис. 39).

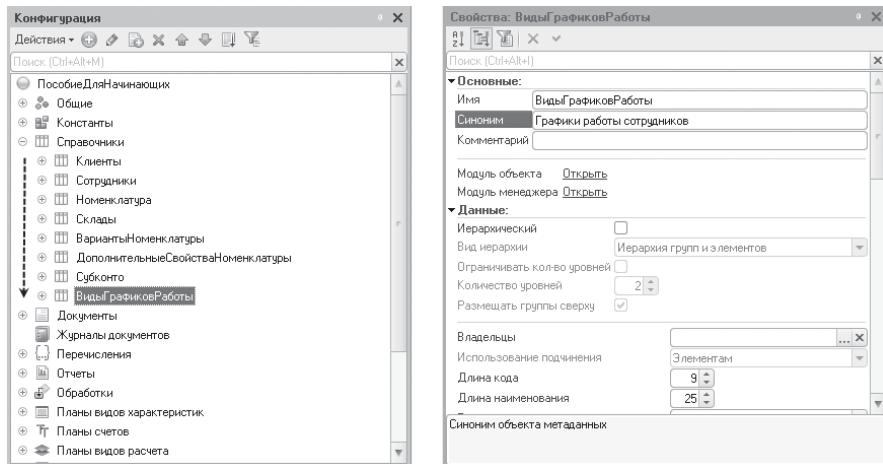


Рис. 39. Палитра свойств текущего объекта конфигурации

Аналогичным образом в палитре свойств можно искать объекты и по комментарию (свойство Комментарий), и по одному из представлений (свойства Представление объекта, Представление списка и т. п.).

Таким образом, для поиска объектов по синониму или представлению (особенно если объектов конфигурации немного или если вам нужно искать только среди нескольких объектов) удобно пользоваться палитрой свойств.

ПРИМЕЧАНИЕ

Если объектов конфигурации много, для поиска объекта по синониму или комментарию (но не представлению) можно также воспользоваться строкой поиска в дереве конфигурации, которая описана в примере «Самый простой способ поиска объектов» на стр. 51. Затем открыть палитру свойств только тех объектов конфигурации, имена которых отображены в дереве черным цветом, и найти среди них нужный объект.

Пример 27. Найти объект по представлению в крупной конфигурации

Предположим, в интерфейсе прикладного решения вы увидели команду для открытия списка документов «Приходные накладные». Скорее всего, это представление списка, потому что имя документа обычно устанавливается в единственном числе («ПриходнаяНакладная»), а автоматически создаваемый синоним документа включает только пробел между словами («Приходная накладная»). То есть название во множественном числе и с пробелом между словами – это, скорее всего, представление списка документов.

То есть вы ищите определенный объект (в нашем примере Документ) по представлению его списка. И предположим, что конфигурация содержит много объектов.

Проблема в том, что свойства Представление объекта, Представление списка и т. п. нельзя найти поиском в окне дерева объектов конфигурации, как показано в примере «Самый простой способ поиска объектов» на стр. 51.

Искать объект с помощью палитры свойств, как было показано в примере «Найти объект по синониму или представлению в небольшой конфигурации» на стр. 54, не совсем удобно, когда конфигурация большая.

Поэтому, в данном случае лучше всего воспользоваться глобальным поиском во всей конфигурации.

Для этого нажмите кнопку Глобальный поиск  на панели инструментов конфигуратора. В окне поиска укажите строку глобального поиска «накладные», снимите все флажки, кроме флашка Свойства (т. к. Представление списка – это свойство) и нажмите Искать или Enter (рис. 40).

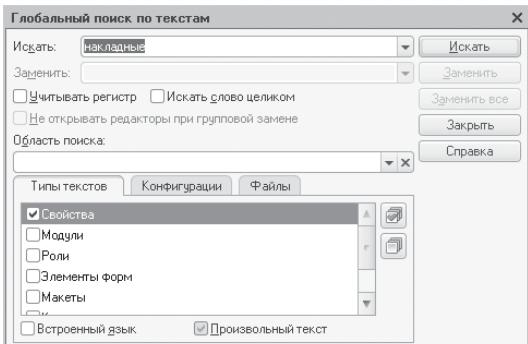


Рис. 40. Окно глобального поиска в конфигурации

После этого откроется окно результатов поиска со списком всех объектов конфигурации, содержащих в любом свойстве строку «накладные». С помощью двойного щелчка мыши на этом списке можно перейти к нужному объекту в дереве конфигурации (рис. 41).

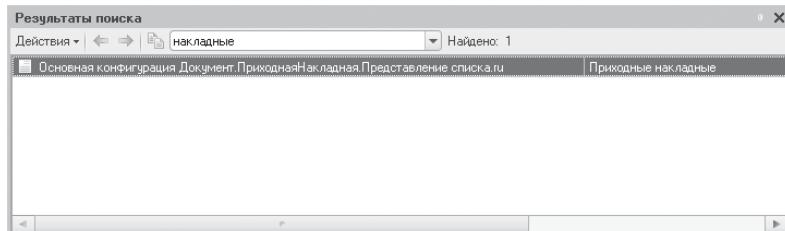


Рис. 41. Окно результатов глобального поиска

Ширина колонок в окне результатов поиска можно регулировать с помощью мыши, удерживая клавишу Ctrl, потянув мышью вертикальную разделительную полосу между колонками списка.

Глобальный поиск может быть очень полезен в случае, когда нужно найти искомую строку сразу на нескольких языках, определенных в конфигурации. Дело в том, что глобальный поиск выполняется во всех строках на разных языках (в данном случае на рис. 41 указано, что искомая строка найдена в представлении списка на языке «ги»). В то время как в палитре свойств мы видим только те строки (например, представление списка), которые соответствуют текущему языку конфигурации (Конфигурация > Язык редактирования конфигурации).

С помощью глобального поиска можно найти практически все. В случае если список результатов поиска слишком большой, то для быстрой ориентации в нем можно использовать уточняющий поиск по Ctrl + F.

ПРИМЕЧАНИЕ

Если объектов конфигурации немного, также хорошим решением может быть поиск объекта по представлению с помощью палитры его свойств, как было описано в разделе «Найти объект по синониму или представлению в небольшой конфигурации».

Пример 28. Найти объект по алфавиту

Для разных конфигураций может быть удобен разный порядок расположения объектов в дереве. Например, если конфигурация небольшая, то может быть удобным в начале расположить «главные» справочники, а в конце – «вспомогательные». А если конфигурация большая, в ней много (100–200) справочников, то может быть удобнее расположить справочники по алфавиту.

В последнем случае можно автоматически отсортировать объекты (например, справочники) по имени в текущей ветви конфигурации.

Для этого нужно раскрыть (или просто выделить) в дереве объектов конфигурации нужную ветвь объектов (в нашем случае, Справочники), выделить ее и нажать кнопку Упорядочить список  в командной панели окна дерева объектов конфигурации. В появившемся окне оставить включенным флажок Сортировать по имени (рис. 42).

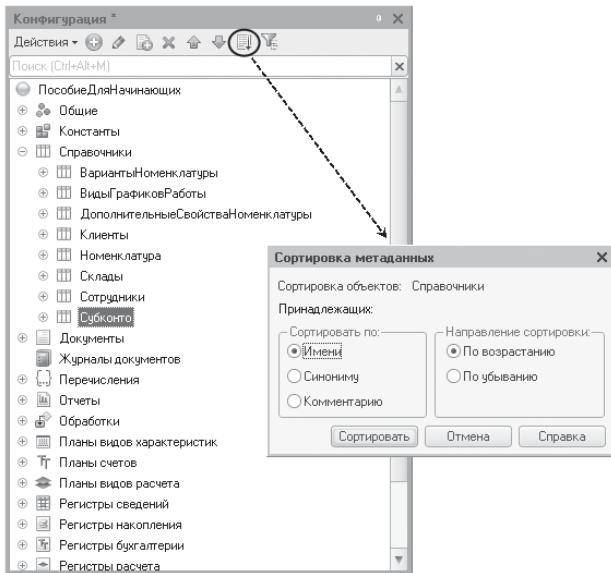


Рис. 42. Сортировка объектов в дереве объектов конфигурации

ПРИМЕЧАНИЕ

Также сортировку объектов конфигурации можно выполнить с помощью команды Действия > Сортировать в командной панели окна объектов конфигурации.

Однако не стоит спешить с этим, казалось бы, очевидным решением вопроса.

Нужно иметь в виду, что сортировка объектов в дереве конфигурации – это необратимая операция. То есть если объекты конфигурации были расположены в некотором «смысловом» порядке, то после выполнения сортировки и сохранения конфигурации вернуть этот порядок (просто отменив сортировку) будет уже нельзя. Поэтому прежде чем выполнять сортировку, нужно быть уверенным, что это не навредит в дальнейшем вам или вашим коллегам.

Если вы все-таки нажали на кнопку Упорядочить список , то, пока вы не обновили конфигурацию базы данных (F7), можно вернуться к той конфигурации, которая осталась в базе данных (Конфигурация > Конфигурация базы данных > Вернуться к конфигурации БД). Если после сортировки конфигурацию базы данных уже обновляли, то тогда, чтобы восстановить прежний порядок объектов в дереве, потребуется загрузить резервную копию конфигурации.

Кроме того, автоматическая сортировка объектов в некоторых случаях может быть просто вредна. Например, можно отсортировать не только сами объекты конфигурации, но и их реквизиты.

Если информационная база участвует в автоматическом обмене данными, то порядок реквизитов в объектах, обменивающихся данными, в обеих базах должен совпадать. Если в одной базе реквизиты объекта отсортированы по алфавиту, а в другой – нет, то стандартный обмен данными между такими базами работать не будет.

Индексы, создаваемые системой, также зависят от порядка реквизитов. Изменение порядка реквизитов в объектах может привести к построению неэффективных индексов, отрицательно сказаться на скорости доступа к данным и быстродействии системы в целом.

Поэтому – совет! Реквизиты лучше не сортировать. Остальное – дело вкуса.

Пример 29. Самый простой способ поиска свойств

Предположим, вы создаете или редактируете форму какого-то объекта и вам нужно установить какие-то свойства элементов формы. Для этого сначала найдите интересующее вас свойство в палитре свойств нужного элемента.

Но перед этим желательно убедиться, что в командной панели палитры свойств отключен показ только важных с точки зрения платформы свойств (кнопка Показывать только важные  не нажата).

Иногда в процессе знакомства с конфигуратором начинающие разработчики включают этот фильтр и забывают о том, что он включен. При включенном фильтре список свойств сокращается до самых важных, и поэтому некоторые разработчики никак не могут найти то свойство, которое им нужно.

Также следует иметь в виду, что в палитре свойств могут отображаться как имена, так и представления свойств. Они могут отличаться друг от друга не только наличием пробелов между словами, но и написанием, например, свойство имеет имя ПутьКДанным, а представление Данные. То есть нужно определить, по чему вы будете искать свойство – по имени или по представлению, и в соответствии с этим установить соответствующий режим в палитре свойств.

Для этого на свободном месте палитры нужно вызвать контекстное меню и установить/снять флажок Отображать имена свойств (рис. 43).

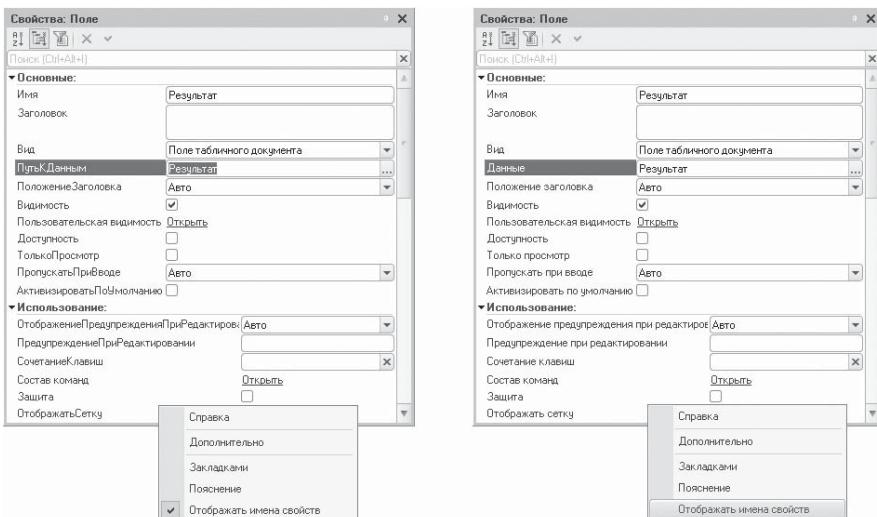


Рис. 43. Режим отображения имен/представлений свойств

Если вы не уверены, имена или представления отображаются в палитре свойств, можно вызвать контекстное меню палитры и проверить состояние флагка Отображать имена свойств. Также об этом можно догадаться по наличию/отсутствию пробела между словами при отображении свойств (ТолькоПросмотр – имя, Только просмотр – представление свойства).

После того как вы поняли, в каком состоянии находится палитра свойств, можно в ней что-нибудь найти.

Предположим, вам нужно установить свойство для элемента формы, которую вы редактируете. Допустим, что вы не помните точно имя свойства, помните только, что оно начинается со слова «отображать» или «отображение». И предположим, что вы с трудом ориентируетесь в палитре свойств, поэтому не знаете, в какой категории свойств искать нужное вам свойство.

Проблема в том, что свойств в палитре довольно много, и искать «глазами» неизвестное свойство довольно утомительно и неэффективно. Поэтому в данном случае удобнее и проще всего использовать для поиска нужного свойства строку поиска в палитре свойств.

В строке поиска нужно ввести искомые символы или фрагменты слов, разделенные пробелом, которые нужно найти в свойстве объекта. В зависимости от режима отображения свойств в палитре (имена или представления) поиск будет производиться либо в именах, либо в представлениях свойств.

Откройте палитру свойств нужного элемента формы, установите курсор в строку поиска вверху окна и введите символы «отобраз». По мере набора символов поиск будет выполняться автоматически. Искомые символы будут подсвечены в найденных свойствах, которые будут представлены обычным списком, без разбивки на закладки и категории (рис. 44).

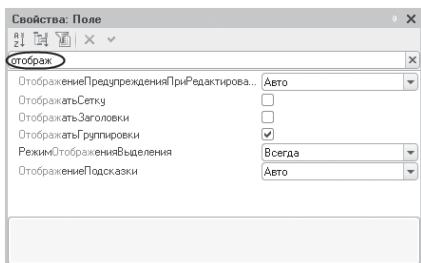


Рис. 44. Поиск свойств в строке поиска палитры свойств

Среди этого списка вы легко догадаетесь, какое свойство вам нужно (например, ОтображениеПодсказки).

В нашем случае все равно, как будет представлена палитра свойств. Так как искомая строка «отобраз» – неполное слово, то оно будет найдено и в именах, и в представлениях свойств. Если же в строку поиска палитры свойств ввести «отображать под» и при этом в палитре отображаются имена свойств, то искомая строка найдена не будет, так как включает пробел, который используется только в представлениях свойств. Но это не страшно. Уже после того, как выполнен поиск, вы можете переключить представление палитры свойств (Отображать имена свойств) и тут же увидеть, есть ли искомая строка в представлениях.

ПРИМЕЧАНИЕ

Чтобы быстро перейти в строку поиска в палитре свойств, можно нажать **Ctrl + Alt + I**.

Пример 30. Найти свойство по алфавиту

Предположим, вам нужно сделать активным по умолчанию какой-то элемент формы при ее открытии. Для этого в палитре свойств этого элемента формы нужно поставить флажок в свойстве АктивизироватьПоУмолчанию.

Стандартно свойства в палитре группируются по категориям Основные, Использование, Оформление и т.д. (при этом кнопка Сортировка по категориям нажата).

Но, предположим, вы не знаете, в какой категории свойств находится нужное вам свойство.

Учитывая это и то, что имя свойства начинается с буквы «A», наиболее эффективным решением будет сортировка всех свойств в палитре по алфавиту. При этом нужное свойство окажется вверху списка и будет на виду. В результате, не прокручивая окно палитры, можно будет быстро найти его визуально.

Для этого нажмите кнопку Сортировка по алфавиту на командной панели окна палитры свойств (рис. 45).

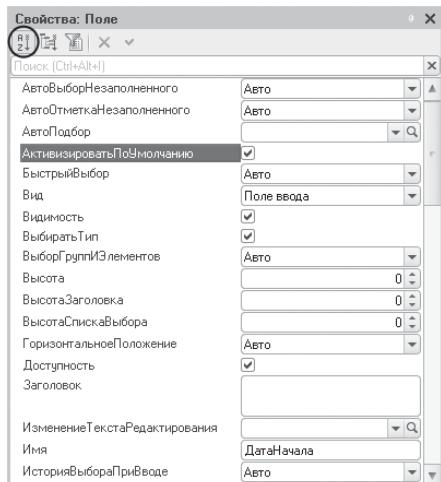


Рис. 45. Сортировка по алфавиту свойств в палитре

Таким образом, когда вы знаете имя свойства, но не уверены, в какой категории свойств его искать, бывает удобно отсортировать свойства в палитре по алфавиту и найти нужное свойство визуально.

ПРИМЕЧАНИЕ

Если имя свойства находится в конце алфавита, то вместо прокручивания палитры свойств можно воспользоваться сочетанием клавиш Ctrl + End.

Пример 31. Подсвечивать найденные фрагменты в тексте программы

Предположим, вам нужно найти использование идентификатора, имя которого вы знаете, но в данный момент его нет на экране.

В этом случае можно, конечно, найти искомую строку с помощью обычного поиска (Ctrl + F), а дальше передвигаться по результатам поиска с помощью клавиш F3/Shift + F3. Но когда найденных слов много, это долго и утомительно.

Гораздо быстрее и проще найти нужное вхождение искомой строки визуально, пролистав текст модуля, когда все результаты поиска выделяются каким-то цветом фона.

Для этого в настройках параметров конфигуратора (Сервис > Параметры > Модули > Редактирование) в поле Цвет фона результатов поиска настройте выделение результатов поиска сиреневым цветом фона (рис. 46).

ПРИМЕЧАНИЕ

Стандартно результаты поиска в модуле выделяются светлым серо-коричневым цветом фона. Если для вас это достаточно ярко, то можно и не выделять результаты поиска особым цветом фона.

Для поиска в тексте модуля, как обычно, нажмите кнопку Поиск текста  (Ctrl + F) в панели инструментов конфигуратора. В появившемся окне введите строку поиска и нажмите Искать или Enter.

Например, вам нужно увидеть, где в модуле используется подстрока «Материал». В результате все места в тексте модуля, где используется искомая подстрока, будут выделены сиреневым цветом фона, и их можно легко охватить одним взглядом (рис. 47).

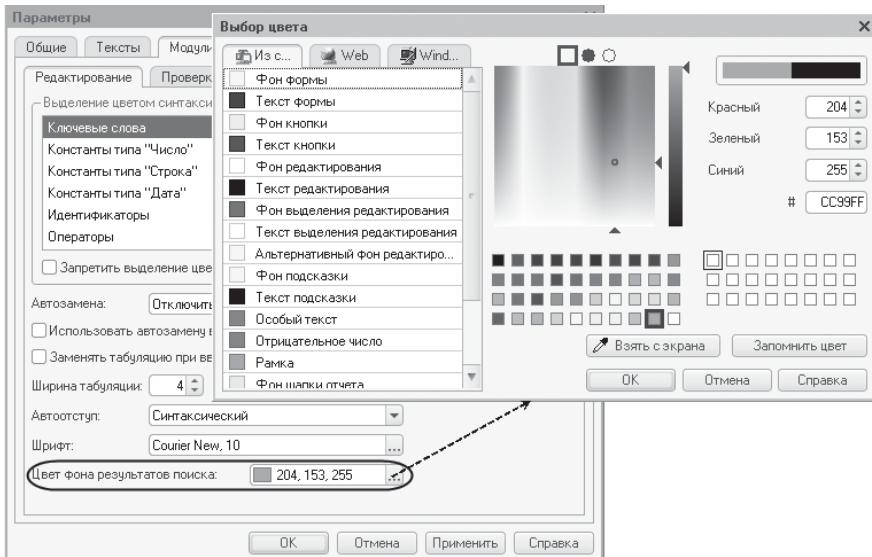


Рис. 46. Настройка подсветки результатов поиска

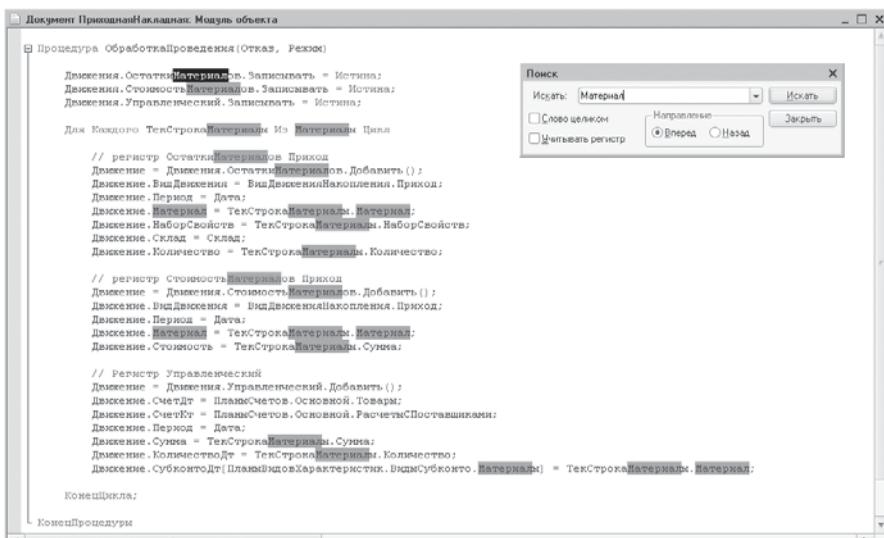


Рис. 47. Выделение цветом результатов поиска в тексте модуля

Пример 32. Найти свойство или метод в синтакс-помощнике

Предположим, вы хотите установить родительский элемент для некоторого плана видов характеристик. Вы знаете, что вам нужно задать свойство Родитель, но сначала хотите прочитать информацию об этом свойстве плана видов характеристик в синтакс-помощнике.

Если искать это свойство в дереве синтакс-помощника, надо хорошо представлять себе структуру этого дерева (Прикладные объекты > Планы видов характеристик > ПланВидовХарактеристикСсылка > Свойства). И даже если вы хорошо ориентируетесь в нем, все равно искать что-то на ветвях дерева – долго и неудобно. Легче всего сделать это поиском по индексу синтакс-помощника.

Чтобы открыть синтакс-помощник, нажмем кнопку Синтакс-помощник (Ctrl + Shift + F1) на панели инструментов конфигуратора. В открывшемся окне перейдем на закладку Индекс, наберем в строке поиска «роди» и двойным щелчком мыши выберем из списка результатов поиска строку «Родитель» (рис. 48).

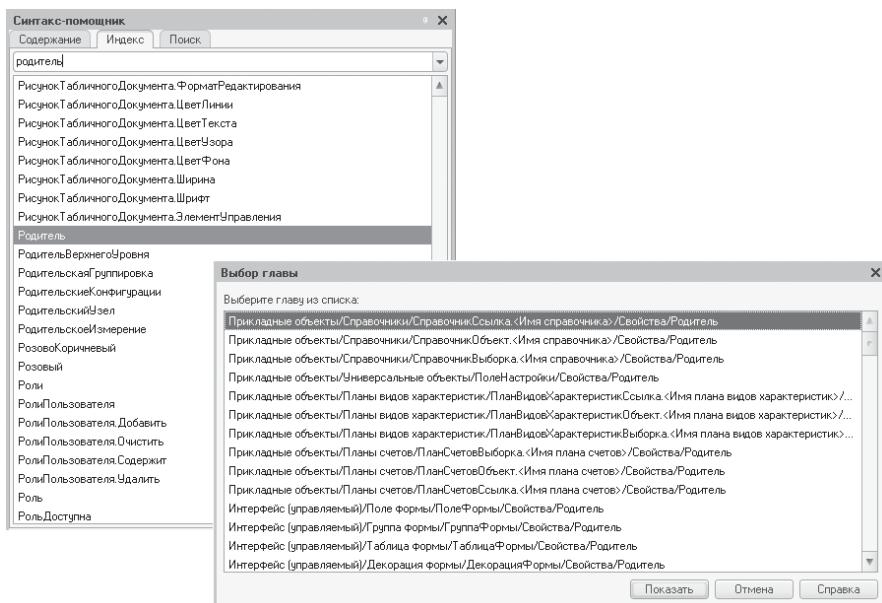


Рис. 48. Поиск по индексу синтакс-помощника

Сначала откроется окно, содержащее список результатов поиска, так как в синтакс-помощнике будет найдено несколько глав с описанием искомого выражения. В этом окне нужно выбрать подходящую главу.

Каждая глава в списке содержит полный путь к искомому выражению, начиная от корня структуры содержания синтакс-помощника. Уровни этой структуры отделены друг от друга символом «/» (слеш). Если длинные названия не видны полностью, вы можете раздвинуть окно в ширину.

Однако часто бывает (как и в нашем случае), что список глав, в которых используется искомое выражение, достаточно большой, и визуально искать нужную главу довольно сложно. В этом случае можно воспользоваться стандартным окном поиска (**Ctrl + F**).

Нам нужно свойство Родитель для плана видов характеристик. Поэтому, находясь в окне списка, нажмем **Ctrl + F** и введем в окно поиска выражение «характер». Нажмем кнопку Искать. После этого в списке глав будет выделена первая глава с найденным выражением (рис. 49).

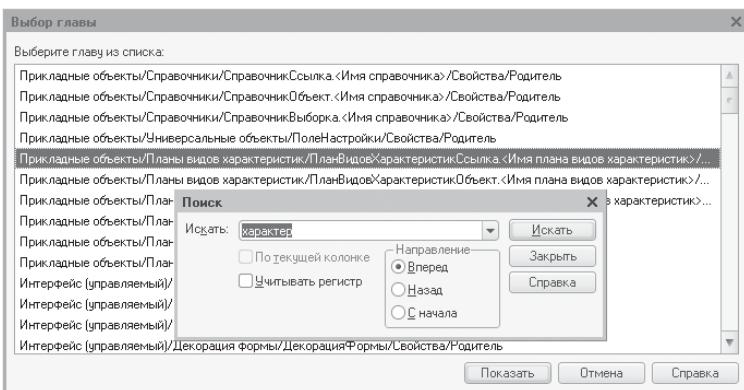


Рис. 49. Поиск нужной главы синтакс-помощника

Нажмем кнопку Показать. Описание свойства Родитель для плана видов характеристик откроется в нижнем окне синтакс-помощника.

Кстати, на закладке Индекс все введенные вами ранее строки поиска запоминаются и предлагаются затем для выбора в выпадающем списке под строкой поиска. Таким образом, с помощью этого списка можно быстро перейти к найденной ранее информации (рис. 50).

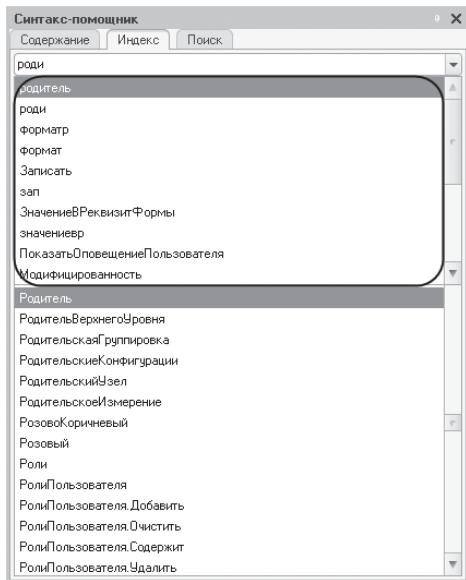


Рис. 50. Выпадающий список у строки поиска по индексу синтакс-помощника

Пример 33. Что написано в сообщении об ошибке

При возникновении ошибки (независимо от того, когда она произошла – при синтаксическом контроле модуля или при его исполнении) в подавляющем большинстве случаев платформа выведет сообщение об ошибке, в котором будет указано место и причина возникновения ошибки (рис. 51).

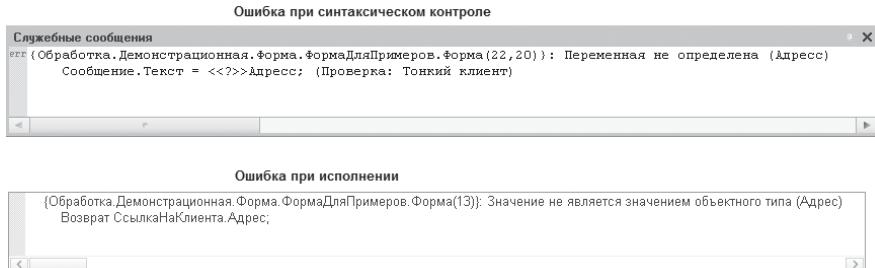


Рис. 51. Сообщения об ошибках

Текст сообщения об ошибке содержит точный путь к расположению ошибки – Обработка.Демонстрационная.Форма.ФормаДляПримеров.Форма, где:

- Обработка – это класс объектов конфигурации (справочник, документ и т. д.);
- Демонстрационная – это имя объекта конфигурации;
- Форма – это указание на коллекцию форм, которые имеются у объекта;
- ФормаДляПримеров – это имя одной из форм объекта;
- Форма – это имя модуля формы.

То есть из сообщения понятно, что ошибка произошла в модуле формы обработки с именем ФормаДляПримеров.

Затем в скобках указан номер строки (22) и номер колонки (20), в которой была найдена ошибка. Если ошибка возникла во время выполнения приложения, то в сообщении указывается только номер строки модуля (13).

Далее следует описание причины возникновения ошибки, например, «Переменная не определена (Адресс)».

Затем приводится строка кода, в которой содержится ошибка.

В случае если ошибка возникла во время синтаксического контроля модуля, предполагаемое место ошибки (в данном случае перед началом имени неопознанной переменной) помечается символами <<?>>. А также указывается, при проверке какого режима исполнения была найдена ошибка. Поскольку модуль формы компилируется и на сервере, и на клиенте, то проверка проходит в два прохода – сначала то, что помечено директивой исполнения &НаСервере и &НаСервереБезКонтекста, потом – &НаКлиенте.

Пример 34. Быстро перейти к строке, в которой ошибка

После того как вы прочитали сообщение об ошибке и поняли, где она возникает, вам нужно найти это место в конфигурации. В большинстве случаев искать это место вручную не нужно, можно перейти к нему автоматически.

Ошибка при проверке модуля в конфигураторе. Если ошибка обнаружена при синтаксическом контроле модуля, то в нижней части экрана появится окно служебных сообщений со списком ошибок. Чтобы быстро перейти к строке модуля, содержащей ошибку, достаточно дважды щелкнуть мышью по сообщению об ошибке. При этом если процедура модуля, в которой найдена ошибка, закрыта, то она будет развернута, и курсор будет установлен на строку с ошибкой (рис. 52).

При этом вы вернетесь в конфигуратор, в котором будет открыт модуль с ошибкой, и курсор будет установлен на строку процедуры, в которой содержится ошибка.

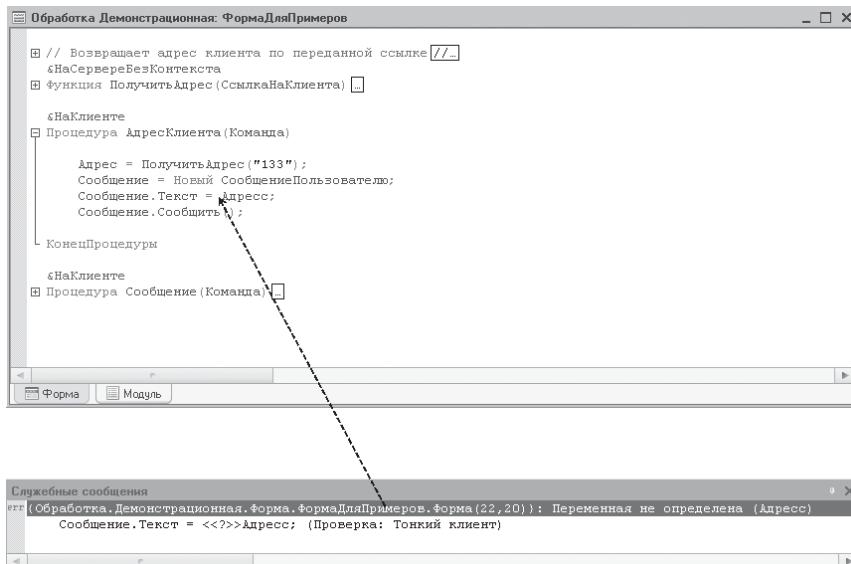


Рис. 52. Переход к строке с ошибкой из окна сообщения об ошибках

Ошибка в режиме «1С:Предприятие», когда приложение запущено в режиме отладки. Если «1С:Предприятие» было запущено в режиме отладки из конфигуратора и во время его выполнения возникла ошибка, то в окне сообщения об ошибке нужно нажать кнопку Подробно и затем перейти в конфигуратор, нажав кнопку Конфигуратор (рис. 53).

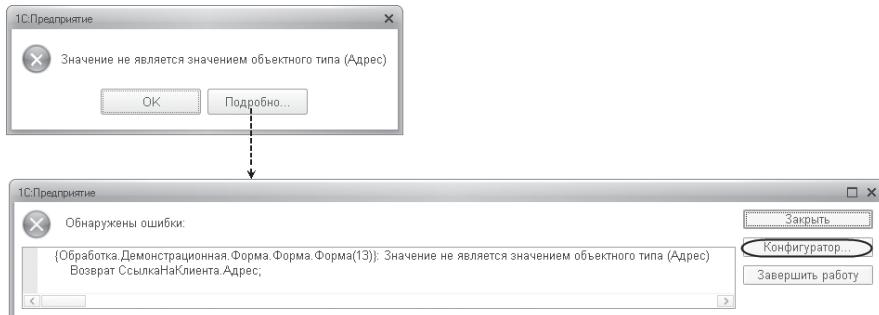


Рис. 53. Переход в конфигуратор из окна подробной информации об ошибке в режиме «1С:Предприятие»

Ошибка в режиме «1С:Предприятие», когда приложение запущено без отладки. Если «1С:Предприятие» было запущено в обычном режиме, без отладки (именно так его запускают пользователи), то после возникновения ошибки вам остается только скопировать полное сообщение об ошибке, а потом «вручную» найти это место в конфигурации.

Пример 35. Найти строку, про которую сказано в сообщении об ошибке

Если у вас есть сообщение об ошибке и нужно вручную найти это место в программе, то сделать это довольно просто. Сначала вам надо открыть модуль, о котором говорится в сообщении, а затем найти в нем строку и, если она указана, колонку, в которой программа обнаружила ошибку.

Конфигурация может содержать много разных модулей, и не все они находятся и открываются одинаково.

Если вам нужен какой-то из модулей, относящихся ко всей конфигурации в целом (модуль управляемого приложения, модуль сеанса, модуль внешнего соединения), то проще всего открыть их из контекстного меню в корне конфигурации (рис. 54).

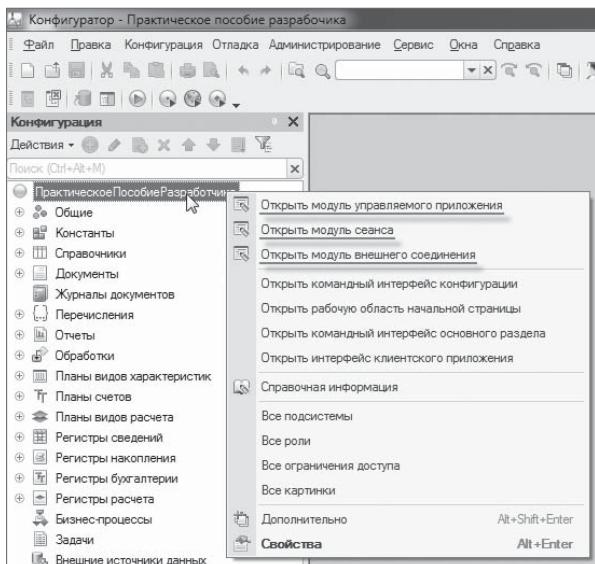


Рис. 54. Модули в корне конфигурации

Если вам нужен какой-то общий модуль, то они находятся в ветке Общие > Общие модули (рис. 55).

Если вам нужны модули какого-то объекта конфигурации (например, модуль объекта или модуль менеджера справочника) или модуль набора записей регистра, то эти модули проще всего открыть из контекстного меню на этом самом объекте конфигурации (рис. 56).

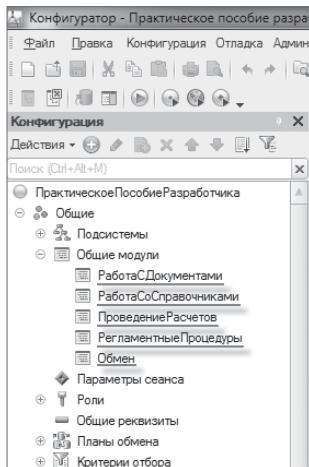


Рис. 55. Общие модули

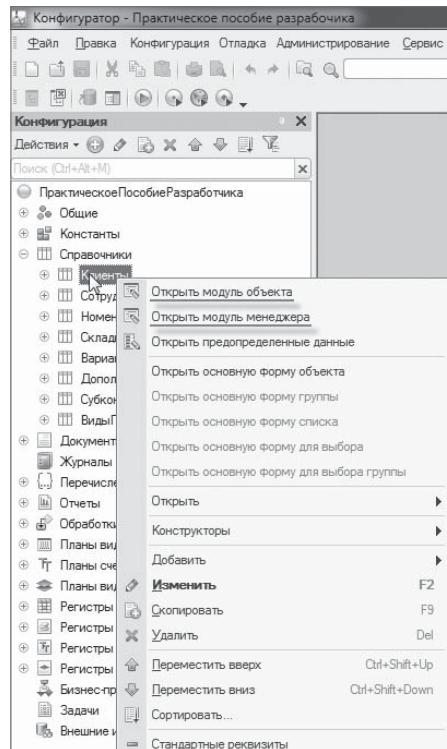


Рис. 56. Модули объектов

Если же вам нужен модуль некоторой формы, то сначала нужно открыть саму эту форму, а затем перейти на ее закладку Модуль (рис. 57).

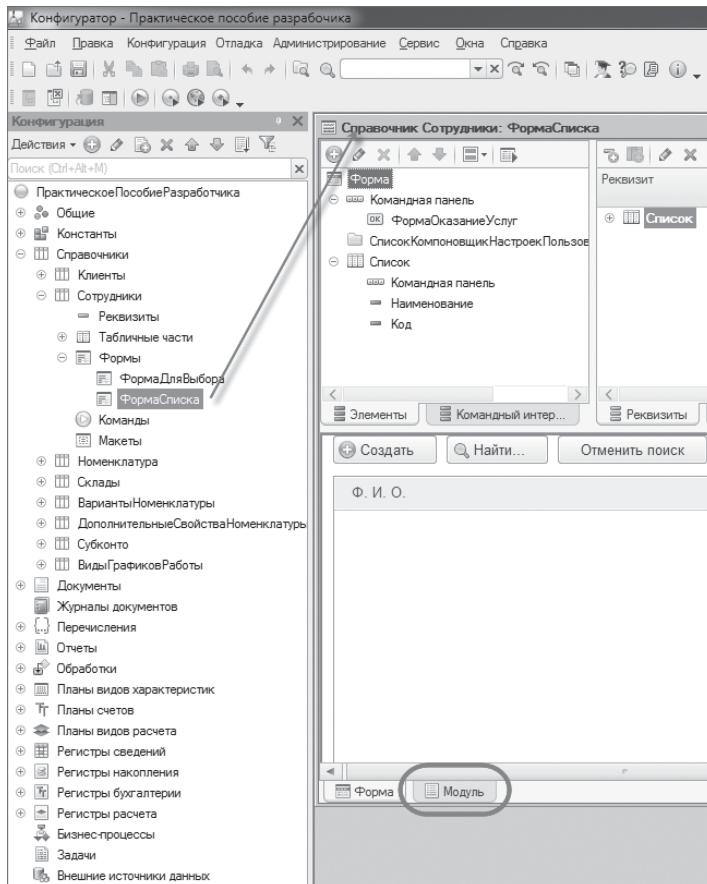


Рис. 57. Модуль формы

После того как вы оказались в нужном модуле, нет необходимости считать строки одну за другой, чтобы найти нужную. Можно быстро перейти к строке с помощью команды Переход к строке... (Ctrl + G), рис. 58.

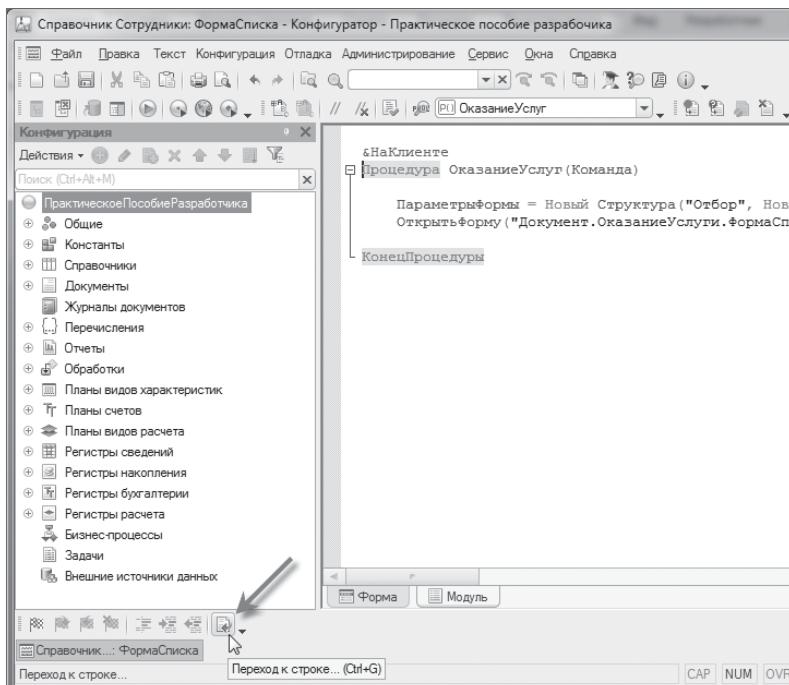


Рис. 58. Переход к строке

Текущая позиция курсора отображается в правом нижнем углу конфигуратора, поэтому если вам нужна какая-то конкретная позиция в этой строке, до нее можно просто пробежать курсором (рис. 59).

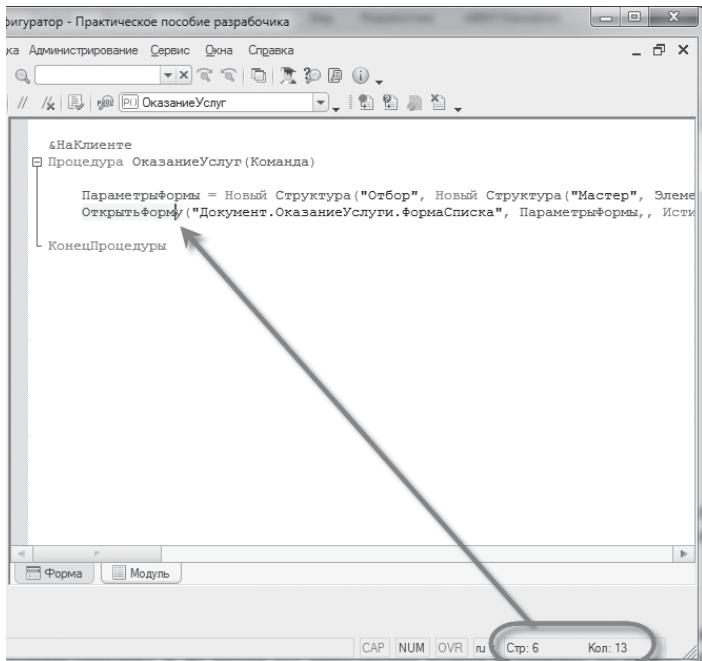


Рис. 59. Позиция курсора

Пример 36. Найти объект конфигурации при выборе

Предположим, вы хотите установить, какие формы будут отображаться на начальной странице прикладного решения.

В этом случае при выборе нужного объекта конфигурации удобно воспользоваться строкой поиска в окне выбора объекта конфигурации. В строке поиска нужно ввести искомые символы, которые требуется найти в имени объекта.

Например, вы хотите установить форму какого-то отчета на начальную страницу приложения. И предположим, что вы не помните точно имя этого отчета.

Поскольку на начальной странице могут отображаться только управляемые формы, то после нажатия кнопки выбора в одной из строк начальной страницы (в окне Рабочая область начальной страницы) откроется окно Выбор управляемой формы. В строке поиска вверху этого окна введите строку «отч». После этого платформа отфильтрует список управляемых форм конфигурации, содержащих искомую строку, и выдаст список этих форм для выбора (рис. 60).

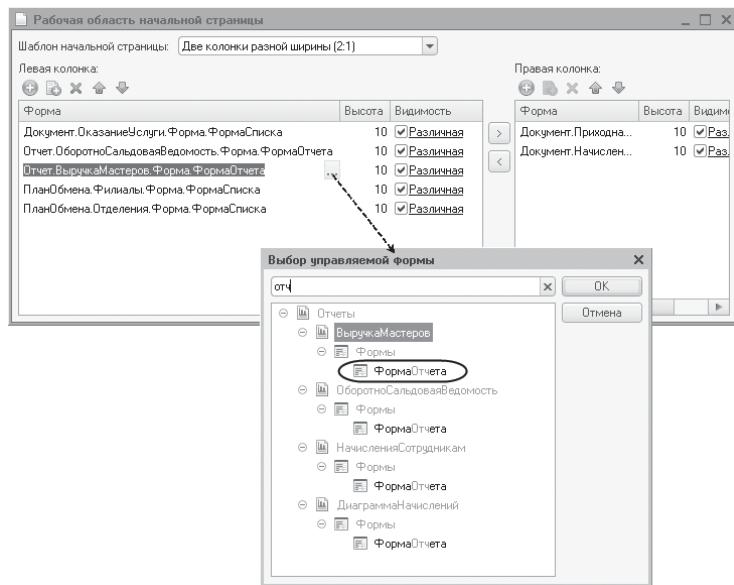


Рис. 60. Поиск форм в окне выбора управляемой формы для заполнения начальной страницы

Серым шрифтом в окне выбора управляемой формы отображены родительские объекты конфигурации – отчеты, имеющие формы, которые можно поместить на родительскую страницу.

Строкаю поиска в окне выбора объекта конфигурации удобно пользоваться также при установке основной формы отчета или при выборе хранилищ настроек, вариантов отчетов и т. п.

ПРИМЕЧАНИЕ

Этим же способом можно пользоваться, когда вам нужно выбрать один из многих типов данных. Например, в палитре свойств или при конструировании реквизита, имеющего составной тип.

Удобное редактирование

Конфигуратор содержит набор полезных возможностей для удобного редактирования текстов процедур и дополнительных свойств объектов. Это также напрямую влияет на скорость и простоту разработки.

Прежде всего, с помощью команд контекстного меню,ываемых непосредственно из программных модулей, вы можете быстро и удобно перемещаться между процедурами модуля, можете найти нужную процедуру модуля по имени, а также легко и просто перейти к определению процедуры, найти использование процедуры в конфигурации и вернуться обратно.

Очень удобно, что прямо из палитры свойств, никуда не отвлекаясь, вы можете перейти к редактированию дополнительных свойств объекта, таких как принадлежность объекта к подсистемам, функциональным опциям и т. п.

Кроме того, в процессе конструирования формы, не запуская конфигурацию, вы можете сразу же оценить, как выглядит разрабатываемая форма в интерфейсе приложения в различных вариантах интерфейса и в различных режимах отображения.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 37. Быстро установить принадлежность к подсистемам

При добавлении и редактировании объектов конфигурации бывает необходимость изменять данные и в палитре свойств, и в окне Дополнительно. Окно Дополнительно предназначено для установки таких свойств объектов конфигурации, как принадлежность объекта к подсистемам, функциональным опциям, планам обмена и т.д. Эти свойства объекта нельзя задать с помощью палитры свойств.

Вообще свойства объекта конфигурации (содержащиеся как в палитре свойств, так и в окне Дополнительно) можно установить в отдельном окне редактирования объекта конфигурации. Но некоторые разработчики не любят им пользоваться, а предпочитают работать в палитре.

Например, вам нужно изменить какое-то из свойств справочника Сотрудники. Для этого вы открыли палитру свойств этого справочника. И теперь вам нужно быстро, никуда не отвлекаясь, установить его принадлежность к подсистемам и функциональным опциям.

Возвращаться в дерево объектов конфигурации и открывать оттуда окно Дополнительно из контекстного меню справочника – это неудобно.

Легче и быстрее перейти в это окно прямо из палитры свойств. Для этого вызовите контекстное меню палитры и выберите пункт Дополнительно (рис. 61).

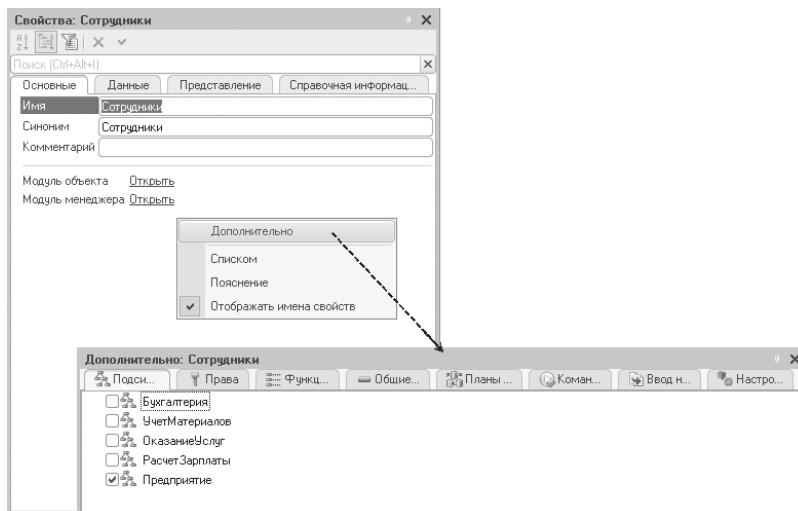


Рис. 61. Вызов окна «Дополнительно» из палитры свойств

Таким образом, возможность устанавливать дополнительные свойства объектов по ходу работы с палитрой свойств очень удобна для тех, кто привык работать исключительно в палитре.

Пример 38. Оценить внешний вид формы

При конструировании формы часто возникает желание посмотреть, как она выглядит в интерфейсе приложения.

Конечно, если вам требуется проверить логику работы и функциональность формы, то нужно запустить сеанс «1С:Предприятия». Но для этого нужно сначала отладить все приложение. Если же вы хотите только оценить внешний вид формы в интерфейсе, то наиболее эффективно это делать сразу в процессе разработки формы в конфигураторе.

Прежде всего, увидеть, как выглядит ваша форма, вы можете в окне предварительного просмотра в нижней части окна редактора формы. Однако изображение формы в редакторе фиксировано по ширине. Понятно, что сложные формы, содержащие много элементов, там не поместятся.

Поэтому если вам хочется посмотреть, как будет выглядеть форма, например, раскрытой на весь экран, воспользуйтесь кнопкой Проверить в командной панели окна элементов формы. После этого вы можете максимизировать окно формы или изменить размеры окна, как требуется (рис. 62).

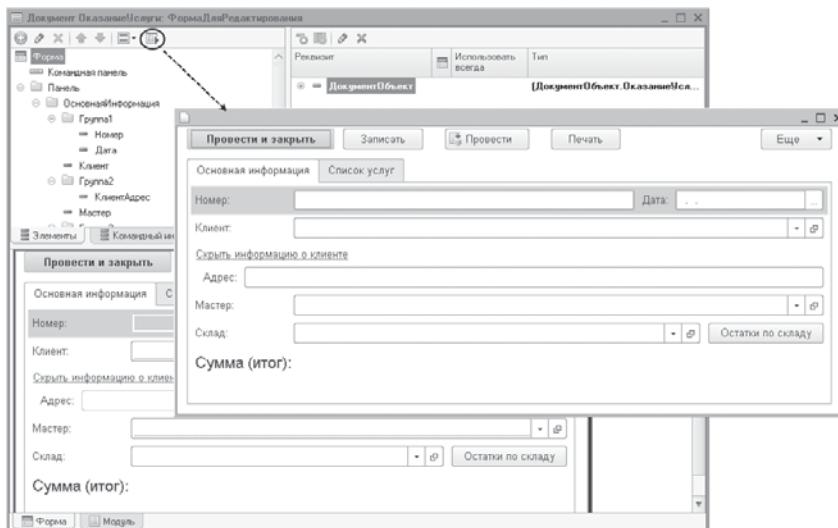


Рис. 62. Вид формы в интерфейсе приложения

Вообще одна и та же форма может иметь несколько «внешних видов». В интерфейсе Такси форму можно представить как в обычном, так и в компактном виде. Кроме того, конфигурация может поддерживать как вариант интерфейса Такси, так и вариант интерфейса Версия 8.2. Конфигуратор позволяет оценить внешний вид формы в любом из этих вариантов.

Для сложных, насыщенных форм, поддерживающих интерфейс Такси, может быть интересно сравнить, как выглядит форма в обычном и компактном режиме. Для этого нажмите кнопку Отображение  в командной панели окна элементов формы и выберите тип отображения Обычное или Компактное (рис. 63).

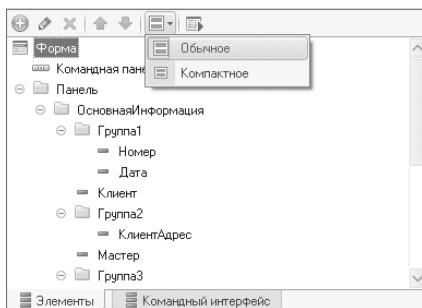


Рис. 63. Переключение режимов отображения в редакторе формы

Кроме этого, с помощью кнопки Вариант интерфейса  в командной панели окна элементов формы вы можете выбрать нужный вариант интерфейса (рис. 64) и оценить, как выглядит форма в интерфейсе Такси (рис. 65) или в интерфейсе Версия 8.2 (рис. 66).

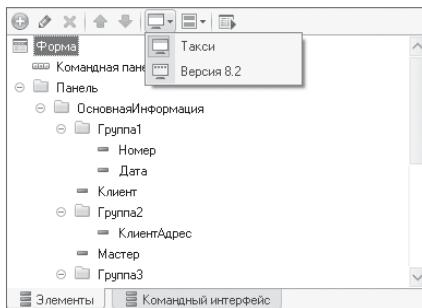


Рис. 64. Выбор варианта интерфейса в редакторе формы

The screenshot shows a window titled 'Провести и закрыть' (Post and Close) at the top left. Below it are two tabs: 'Основная информация' (Main Information) and 'Список услуг' (List of Services). The main area contains several input fields: 'Номер:' (Number), 'Клиент:' (Client), 'Адрес:' (Address), 'Мастер:' (Master), 'Склад:' (Warehouse), and 'Сумма (итог):' (Sum (Total)). There are also buttons for 'Провести' (Post), 'Печать' (Print), and 'Еще' (More) with a dropdown arrow. A 'Скрыть информацию о клиенте' (Hide client information) link is visible above the address field.

Рис. 65. Вид формы в редакторе в интерфейсе «Такси»

This screenshot is identical to Figure 65, showing the same form fields and layout in the 'Taxis' application's interface.

Рис. 66. Вид формы в редакторе в интерфейсе «Версия 8.2»

Наличие кнопок Отображение и Вариант интерфейса в редакторе формы зависит от того, в какое значение установлено свойство конфигурации Режим совместимости интерфейса.

Пример 39. Перейти к известной процедуре модуля

Предположим, вы редактируете форму, в модуле которой содержится большое количество процедур и функций. И вам нужно понять, в каких обработчиках стандартное поведение формы было изменено (путем создания собственных алгоритмов), а в каких оно осталось без изменений.

Просматривать большой модуль визуально – неудобно и неэффективно.

Наиболее просто это сделать с помощью окна Процедуры и функции, вызывающегося кнопкой Процедуры и функции (Ctrl + Alt + P) в панели инструментов конфигуратора.

В открывшемся окне вы увидите список всех процедур и функций модуля. Обработчики событий формы и ее элементов, обработчики команд формы, процедуры и функции, созданные разработчиком. Слева от имени процедур и функций находятся соответствующие пиктограммы. Р() – для процедур, F(x) – для функций. Сначала в этом списке расположены процедуры, написанные разработчиками, а в конце, в угловых скобках, стандартные процедуры, обработчики которых еще не написаны.

При нажатии кнопки Перейти текст выбранной процедуры или функции будет открыт в окне модуля (рис. 67).

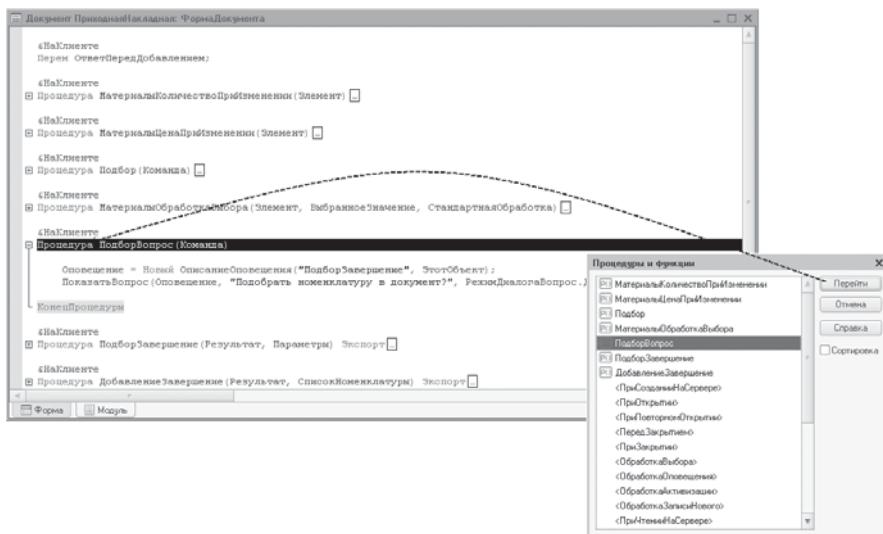


Рис. 67. Окно «Процедуры и функции»

Рассмотрим более конкретную задачу. Например, вас интересует, выполняется какой-то дополнительный код при записи формы или нет.

В этом случае проще всего прокрутить вниз список процедур и функций и посмотреть, написаны ли обработчики интересующих вас событий, или нет. Если среди «пустых» обработчиков интересующего события нет и при этом в модуле находится много разных процедур и функций, можно найти нужную процедуру по имени с помощью Ctrl + F и далее двигаться по результатам поиска клавишей F3 (рис. 68).

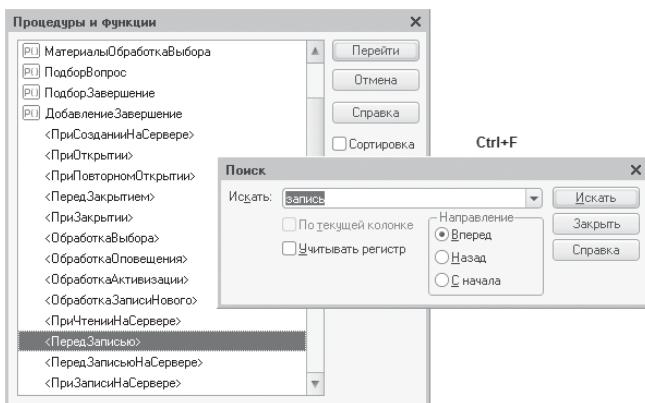


Рис. 68. Выпадающий список процедур и функций модуля

При этом нужно учитывать, что поиск по Ctrl + F ищет, начиная от текущего положения курсора. Поэтому сначала вам нужно встать в начало списка процедур и функций с помощью Ctrl + Home.

Если вы знаете имя нужной вам процедуры, можно установить в окне Процедуры и функции флажок Сортировка, чтобы отсортировать процедуры и функции по алфавиту и легче найти то, что нужно. Стандартно они расположены в списке в порядке их расположения в модуле.

ПРИМЕЧАНИЕ

Если вы просматриваете длинную процедуру и уже не видите на экране ее определение, вы всегда можете узнать имя этой процедуры. Оно показывается в выпадающем списке рядом с кнопкой Процедуры и функции в панели инструментов конфигуратора.

Пример 40. Находясь в месте вызова процедуры, перейти к ее содержимому

Довольно часто вам требуется понять, что происходит в процедуре или функции, в месте вызова которой вы находитесь. Эта процедура может быть создана другими разработчиками или же это может быть ваша собственная функция, написанная довольно давно, и ее нужно модифицировать. Но, сначала в ней нужно разобраться. Для этого вы хотите быстро перейти к определению этой функции. Как сделать это просто и легко?

Искать определение процедуры или функции глобальным поиском по конфигурации долго и неудобно. Зато в конфигураторе есть удобная возможность перехода к определению процедуры или функции из места ее вызова. Причем такой переход возможен не только внутри одного модуля. Переходит можно к определению экспортруемых процедур и функций общих модулей, модулей объектов и модуля приложения.

Предположим, из модуля формы документа вызывается процедура общего модуля РассчитатьСумму() для пересчета данных по строке табличной части документа. В нее передается параметр, содержащий данные строки табличной части. Вы хотите посмотреть и понять, как рассчитываются эти данные.

Для этого установите курсор на место вызова этой процедуры в модуле формы документа, вызовите контекстное меню и выполните пункт Перейти к определению (F12).

Поскольку процедура пересчета определена в общем модуле, перед переходом к ней на экран будет выведен список объектов перехода: экспортруемая процедура общего модуля и сам общий модуль как объект конфигурации. Выберите саму процедуру РассчитатьСумму() и сразу перейдите в тело этой процедуры в общем модуле (рис. 69).

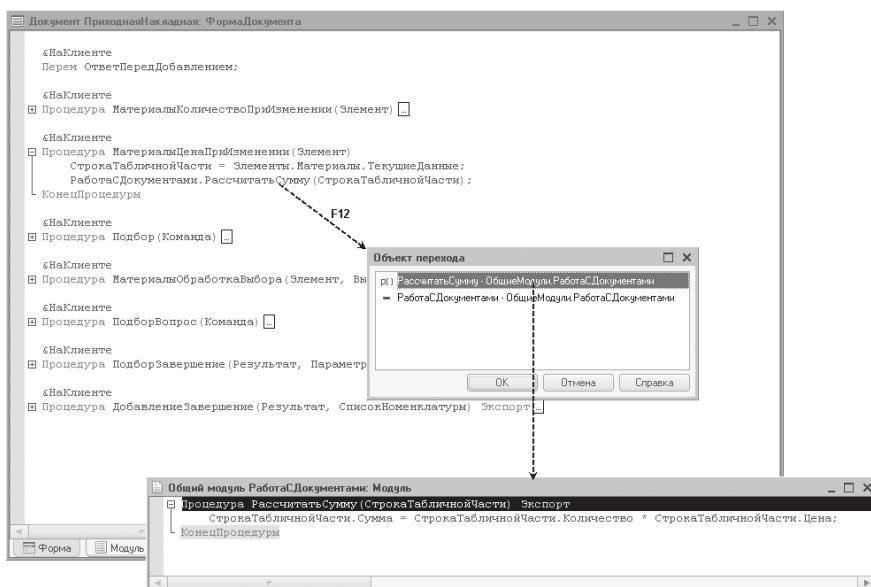


Рис. 69. Переход к определению процедуры или функции

Таким же образом вы можете переходить и к определению переменных.

Для того чтобы вернуться к точке, из которой был осуществлен переход к определению, нажмите сочетание клавиш **Ctrl + -** (рядом с клавишей **=**).

Переход к определению процедур и функций может быть очень полезен также в случае, когда при проверке модуля вы получаете ошибку типа «Слишком много/недостаточно фактических параметров». Для устранения этой ошибки нужно сравнить количество и тип параметров в месте вызова и в месте определения процедуры или функции. Для этого также очень удобно использовать переход к определению по клавише **F12**.

ПРИМЕЧАНИЕ

Для небольших модулей можно использовать и другой способ. Например, вы точно знаете, что вызываемая функция или процедура описывается в том же модуле, откуда вы ее вызываете. Кроме этого, у вас настроено выделение цветом текущих или выбранных идентификаторов. Тогда, выделив имя функции в месте вызова, вы можете визуально найти ее определение и сравнить количество параметров в ее описании и в вашем вызове.

Пример 41. Узнать, в каких модулях используется данная процедура

Например, вы находитесь в теле процедуры общего модуля **РассчитатьСумму()** и хотите ее использовать в модуле формы какого-то документа для пересчета данных табличной части. И, предположим, что процедура не описана согласно рекомендуемым стандартам (не описано ее назначение и тип параметров). В результате вы сомневаетесь, как правильно вызывать эту процедуру, что туда передавать. Поэтому вы хотите посмотреть на пример ее использования.

Можно, конечно, попытаться найти вхождения этой процедуры глобальным поиском по конфигурации, но это – долго и неудобно.

Гораздо проще и быстрее установить курсор на имя процедуры и найти те места конфигурации, откуда она вызывается.

Для этого вызовите контекстное меню и выполните пункт **Найти использование** (**Alt + F12**), рис. 70.

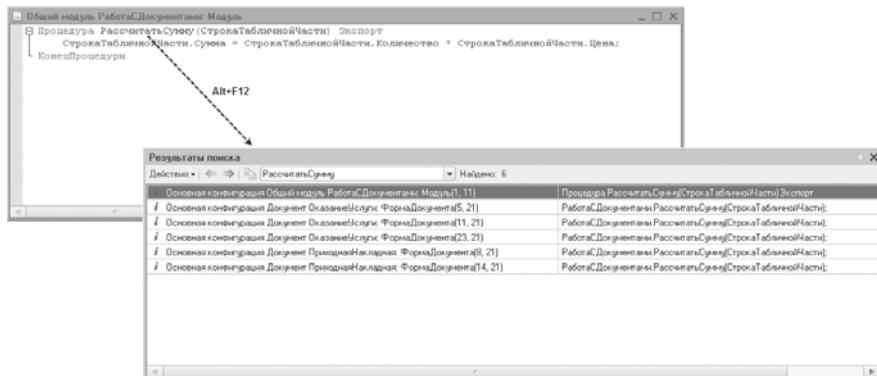


Рис. 70. Поиск использования процедуры или функции

В результате будет открыт список всех модулей конфигурации, откуда эта функция вызывается. Из списка модулей, как обычно, можно двойным щелчком мыши перейти к интересующему вас месту использования функции.

ПРИМЕЧАНИЕ

Для того чтобы вернуться к точке, из которой был осуществлен переход к использованию, нажмите сочетание клавиш **Ctrl + "-"** (рядом с клавишей "=").

Не создавать одно и то же

В данном разделе речь пойдет о том, что не нужно создавать несколько раз одно и то же, когда можно просто скопировать уже созданный ранее объект, реквизит, элемент формы и пр. Копирование не только сэкономит ваше время, но и позволит избавиться от неизбежных ошибок, которые появляются во время дублирования одних и тех же действий.

Вы можете копировать объекты через буфер обмена как в пределах одной, так и между разными конфигурациями. Внутри одной небольшой конфигурации для копирования также удобно использовать перетаскивание.

Таким образом, если вам нужен объект конфигурации, реквизит, элемент формы и пр. и у вас уже есть однотипный объект, то гораздо эффективнее скопировать его и затем лишь изменить его имя и некоторые свойства, чтобы получить то, что нужно.

При редактировании одинаковых свойств у нескольких элементов, команд, параметров или реквизитов формы вы также имеете возможность задать их один раз для всех выделенных объектов, а не повторять все заново для каждого элемента по отдельности.

Кроме того, чтобы не писать имена объектов и методов вручную, проще и быстрее перетащить их из дерева объектов конфигурации или из синтакс-помощника прямо в модуль, тем самым исключив возможность ошибок.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 42. Копировать объекты в небольшой конфигурации

Часто бывает нужно создать в конфигурации несколько похожих друг на друга объектов. Например, вы добавили в конфигурацию приходную накладную. Теперь вам нужно добавить расходную накладную. И тут вы видите, что многие реквизиты у этих документов совпадают, а табличная часть расходной накладной вообще такая же, что и у приходной накладной.

Что же делать? Снова все это создавать еще раз? Нет! Это долго, нудно и чревато ошибками.

В данной ситуации проще всего скопировать полностью тот документ, который вы уже добавили раньше. И затем изменить у появившегося нового документа имя, реквизиты и те свойства, которые присущи новому документу.

Если конфигурация небольшая, эффективнее всего сделать это с помощью перетаскивания мышью исходного документа в родительскую ветвь объектов конфигурации (при этом рядом с указателем мыши появится признак копирования – символ «+»).

Выделите в дереве объектов конфигурации исходный документ **ПриходнаяНакладная** и перетащите его мышью в ветвь **Документы** (рис. 71).

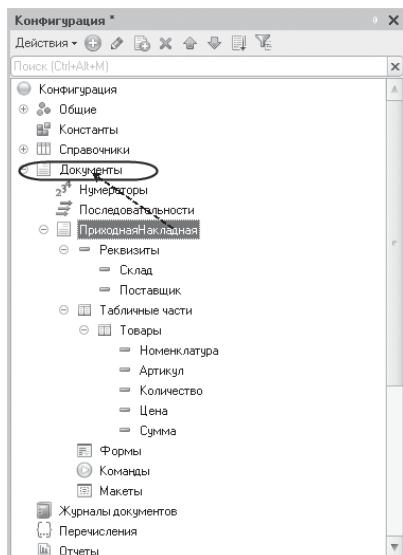


Рис. 71. Копирование объекта в дереве объектов конфигурации

Новый объект будет скопирован в ту же ветвь объектов конфигурации с именем, к которому будет добавлено число копий объекта – 1, 2 и т.д. (*ПриходнаяНакладная1*). Остается только изменить имя объекта (*РасходнаяНакладная*), представление и один из реквизитов (рис. 72).

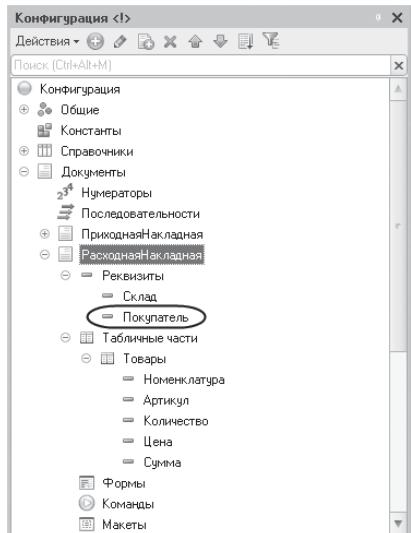


Рис. 72. Копирование объекта в дереве объектов конфигурации

В результате с минимальными трудозатратами вы получите новый документ, созданный путем копирования исходного документа, как если бы вы создавали его с нуля руками.

ПРИМЕЧАНИЕ

Для того чтобы скопировать выделенный документ, вы можете также нажать кнопку Добавить копированием в командной панели окна объектов конфигурации или выполнить команду Скопировать (F9) из контекстного меню документа.

Копировать можно не только целые объекты конфигурации, но и их отдельные реквизиты, табличные части и т. п.

Например, вы уже создали документ *ПриходнаяНакладная*, а теперь надо создать регистр накопления *ОстаткиТоваров*, в котором будут храниться движения этого документа. Понятно, что измерения и ресурсы регистра

будут совпадать по типу с реквизитами документа, значениями которых будут заполняться движения.

Если конфигурация небольшая (т.е. можно, не прокручивая окно дерева, увидеть и реквизиты документа, и измерения или ресурсы регистра), можете просто перетащить мышью реквизиты документа в измерения, ресурсы или реквизиты регистра (рис. 73).

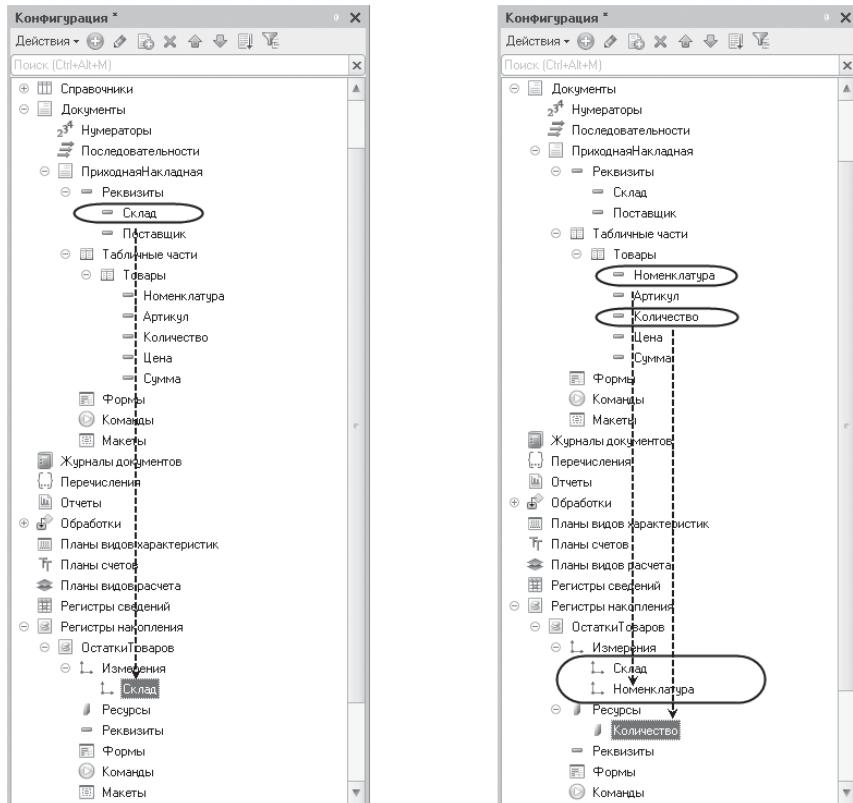


Рис. 73. Копирование реквизитов между объектами конфигурации

В результате быстро и просто вы получите измерения и ресурсы регистра наполнения, совпадающие с соответствующими реквизитами документа и его табличной части, как если бы вы их добавляли в окне редактирования регистра.

Пример 43. Копировать объекты из другой конфигурации

Предположим, вам нужно создать в новой конфигурации какой-то объект, уже существующий в другой конфигурации. Или же вам нужно скопировать объект в пределах достаточно большой конфигурации.

Вы можете сделать это через буфер обмена, стандартными командами Ctrl + C/Ctrl + V.

Рассмотрим пример копирования справочника из одной конфигурации в другую.

Для этого выделите копируемый объект в исходной конфигурации (в нашем примере, справочник Номенклатура), нажмите Ctrl + C, затем выделите в новой конфигурации соответствующую ветвь объектов (Справочники) и нажмите Ctrl + V (рис. 74).

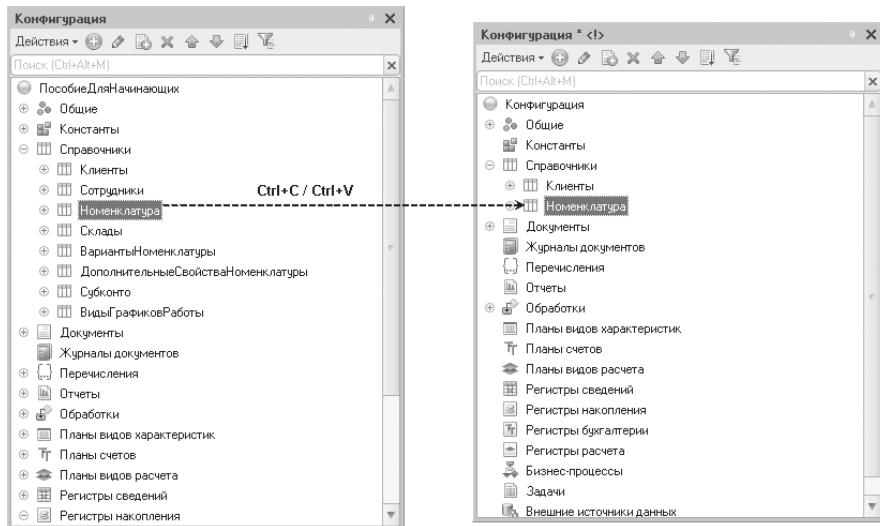


Рис. 74. Копирование объектов между разными конфигурациями

Если при копировании будут обнаружены неразрешимые ссылки (например, в новой конфигурации отсутствует тип, на который ссылается реквизит скопированного объекта), то в окне сообщений будет выдано предупреждение об этом.

Поэтому после копирования нужно проверить свойства нового объекта, установленные платформой, и сделать необходимые исправления. Однако все равно такой способ копирования позволяет значительно облегчить ваш труд. Советуем копировать сначала перечисления, затем справочники, затем документы, на которые они ссылаются, и т. д.

Пример 44. Редактировать сразу несколько реквизитов

Довольно часто бывает нужно установить одинаковые свойства сразу для нескольких элементов формы.

Предположим, в форме обработки вам нужно отключить доступность для кнопки командной панели **ФормаРассчитать** и поля табличного документа **Результат**. А затем при создании формы программным путем доступность этих элементов будет включена при выполнении некоторого условия.

Для этого в палитре свойств каждого из этих элементов вам нужно установить свойство **Доступность** в значение **Ложь**.

Чтобы не открывать отдельно палитру свойств каждого элемента, не искать нужное свойство и не выполнять однотипные действия несколько раз, удобнее всего выделить нужные элементы в дереве формы, открыть общую для них палитру свойств и один раз установить нужное свойство сразу для всех выделенных элементов.

Выделите (с нажатой клавишей **Ctrl**) в дереве формы кнопку командной панели **ФормаРассчитать** и поле табличного документа **Результат** и из контекстного меню одного из них откройте палитру свойств. Заметьте, что в палитре свойств присутствуют только те свойства, которые есть у обоих выделенных элементов. Снимите флажок у свойства **Доступность** (рис. 75).

Подобным образом, особенно если элементов в форме много, удобно устанавливать свойства, отвечающие за отображение сразу для всех выделенных элементов формы.

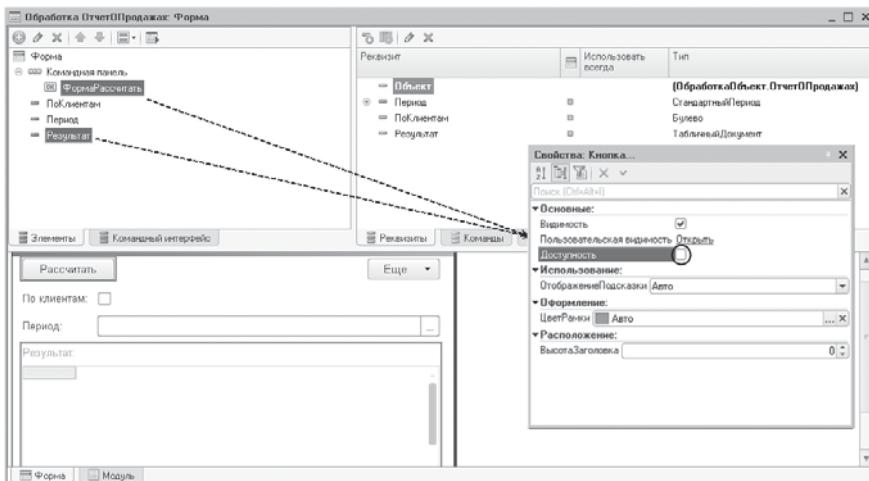


Рис. 75. Установка свойства для нескольких выделенных элементов формы

Пример 45. Копировать реквизиты, команды и элементы

Предположим, в форме обработки вам нужно иметь возможность вводить две произвольные даты, определяющие начало и конец отчетного периода. Допустим, для этого вы уже добавили реквизит **ДатаНачала**, установили тип даты – **Дата** и состав даты – **Дата** (рис. 76).

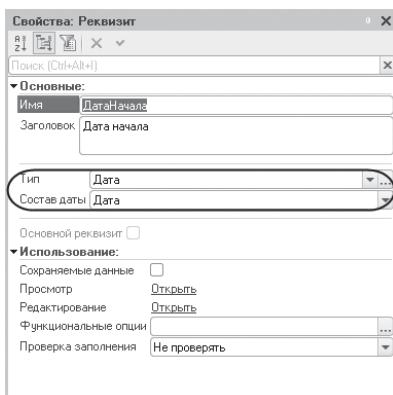


Рис. 76. Палитра свойств реквизита формы «ДатаНачала»

Теперь вам нужно добавить такой же реквизит, но с именем ДатаОкончания.

Чтобы не повторять все заново, вы можете скопировать первый реквизит через буфер обмена, вставить новый реквизит и изменить только имя (и другие необходимые свойства) в его палитре свойств.

Выделите реквизит ДатаНачала в окне реквизитов и перетащите его мышью на свободное место (или просто на соседний реквизит). Новому реквизиту будет присвоено имя ДатаНачала1 (рис. 77).

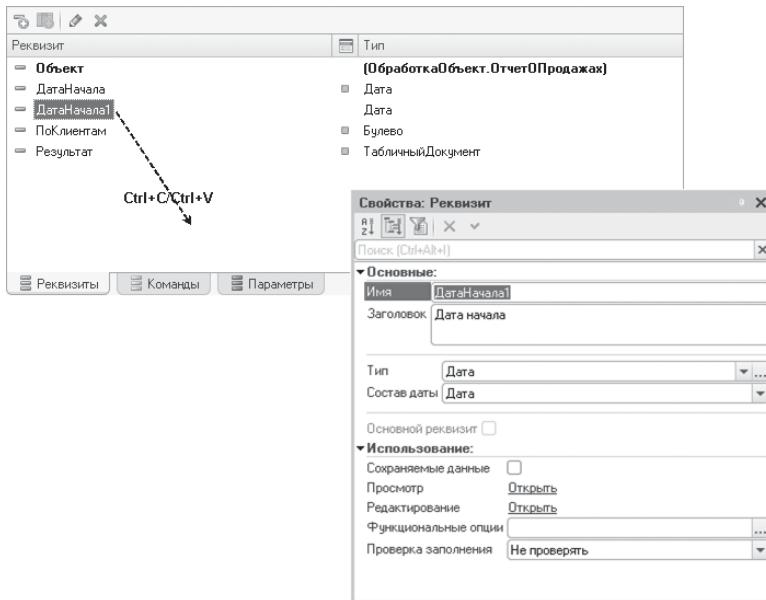


Рис. 77. Копирование реквизитов в редакторе формы

Вам остается только изменить имя реквизита на ДатаОкончания и соответствующим образом подправить заголовок реквизита, так как он остался прежним. Вот и все!

Подобным образом, вы можете копировать также команды и параметры формы, если это нужно.

ПРИМЕЧАНИЕ

Скопировать реквизиты, команды, параметры и элементы формы вы можете также через буфер обмена, командами Ctrl + C/Ctrl + V (для элементов формы возможен только этот способ).

Пример 46. Перетаскивать элементы

Допустим, у вас есть форма, которая содержит много элементов и имеет сложную структуру вложенности групп и страниц друг в друга. И вы хотите изменить расположение элементов в этой форме. Вы можете сделать это «легким движением руки» – перенести группу вместе с ее подчиненными элементами в другое место дерева формы, изменить уровень ее иерархии, превратить страницу в обычную группу и наоборот.

То есть вместо того, чтобы вручную создавать новые группы или изменять свойства какой-то сложной по структуре группы, вы можете сначала перетащить исходную группу в нужное место дерева, а затем открыть ее палитру свойств и посмотреть, как платформа автоматически установила свойства группы. Иногда, возможно, вам понадобится подправить какие-то свойства.

Например, перетащите обычную группу (Группа3) с горизонтальной группировкой в группу страниц (Панель). В результате обычная группа станет еще одной страницей, на которой будут размещены все ее подчиненные элементы (рис. 78, 79).

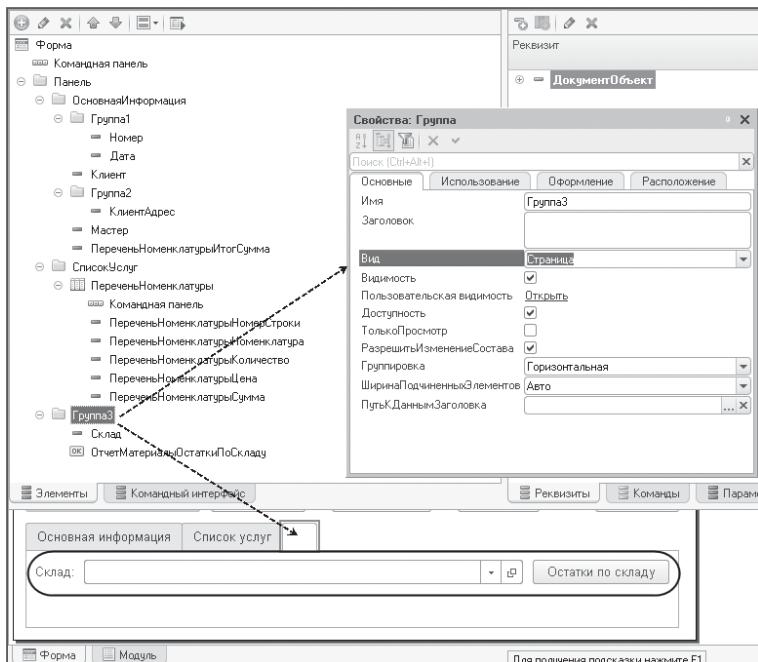


Рис. 78. Перетаскивание элементов формы

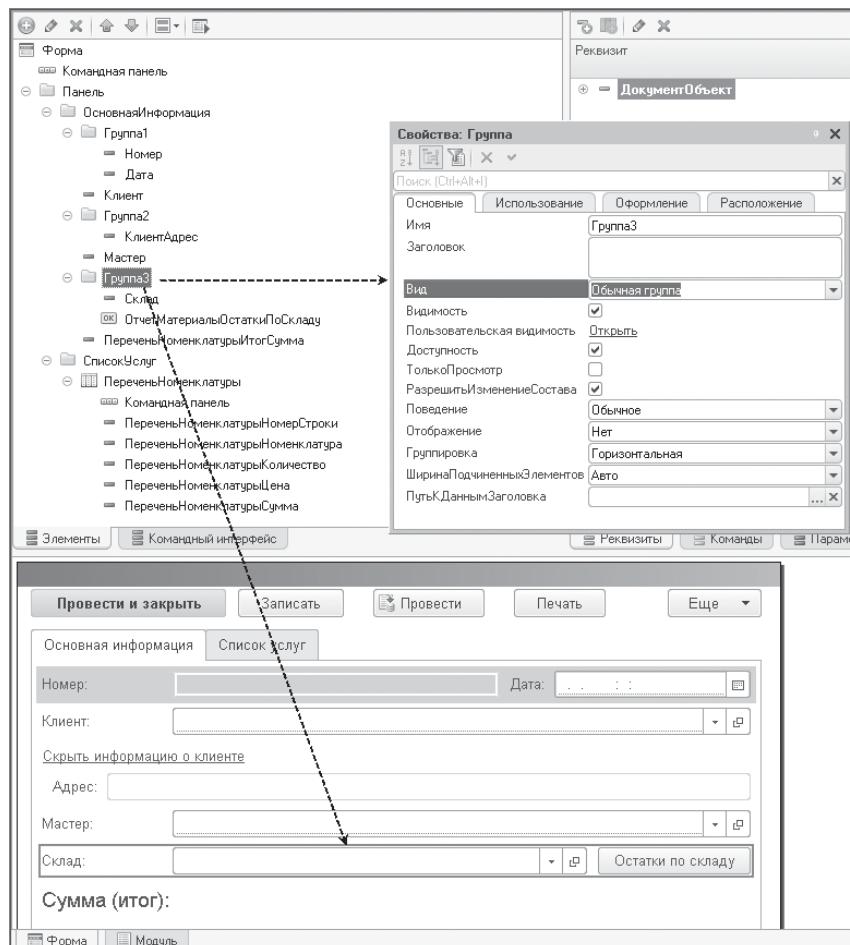


Рис. 79. Перетаскивание элементов формы

Таким образом, вам вообще не пришлось ничего делать вручную, кроме самого перетаскивания.

Если все-таки вдруг «дрогнула рука», вы можете просто откатиться назад, нажав кнопку Отменить в панели инструментов конфигуратора.

Пример 47. Перетаскивать имена объектов в код модуля

Предположим, в модуле объекта вам нужно обратиться к какому-то объекту по имени, например, установить значение реквизита этого объекта.

Чтобы не ошибиться, вы можете не писать его вручную. Лучше всего перетащите его в модуль прямо из дерева объектов конфигурации.

Для этого откройте модуль объекта, затем перейдите в дерево объектов конфигурации, выделите реквизит и просто перетащите его мышью в нужное место модуля (рис. 80).

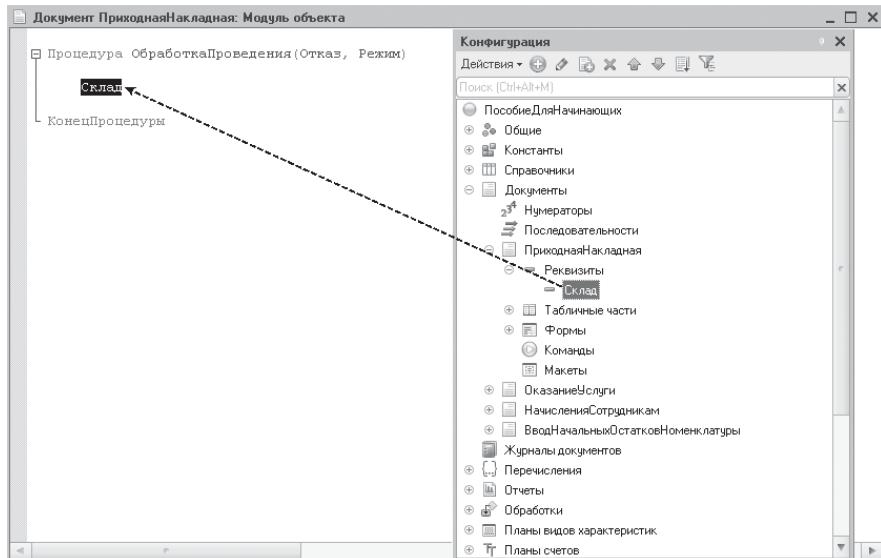


Рис. 80. Перетаскивание имени объекта из дерева объектов конфигурации

После этого вы можете быть уверенными, что имя реквизита написано правильно и что вы не получите ошибки в этом месте при проверке модуля.

ПРИМЕЧАНИЕ

Вы можете также скопировать выделенный в дереве конфигурации объект в буфер обмена ($Ctrl + C$) и затем вставить его имя в текст модуля ($Ctrl + V$).

Пример 48. Перетаскивать методы, конструкторы в код модуля

Часто возникают ситуации, когда вы нашли в синтакс-помощнике нужный вам метод или свойство и хотите его использовать в какой-то процедуре. Чтобы не ошибиться, не нужно писать его руками. Вы можете просто перетащить его в модуль прямо из синтакс-помощника.

Например, вам нужно в форме документа пересчитать данные и сразу же отобразить их пользователю. Для этого вам нужен метод управляемой формы, но какой?

Чтобы внести ясность, откройте синтакс-помощник кнопкой Синтакс-помощник  (Ctrl + Shift + F1) в панели инструментов конфигуратора. Раскройте раздел Интерфейс (управляемый) > Управляемая форма. Затем раскройте объект УправляемаяФорма и его группу Методы. Прочитайте информацию о каждом методе формы в нижней части окна синтакс-помощника.

Из описания понятно, что для решения поставленной задачи вам нужен метод формы ЗначениеВРеквизитФормы(). Имя метода слишком длинное, и если писать его «руками», велика вероятность ошибок.

Поэтому лучше всего перетащить его в модуль прямо из окна синтакс-помощника (рис. 81).

После этого в модуль попадет текст – ЗначениеВРеквизитФормы(,);. При этом можно быть уверенными, что ошибки в полученном тексте нет.

Но, как выяснилось, вы не помните, что нужно передавать в этот метод. Поэтому сразу после перетаскивания нажмите Ctrl + Shift + <Пробел>. При этом откроется окно контекстной подсказки параметров метода (рис. 82).

ПРИМЕЧАНИЕ

Вы можете также скопировать выделенную в дереве синтакс-помощника строку в буфер обмена (Ctrl + C) и затем вставить ее в текст модуля (Ctrl + V).

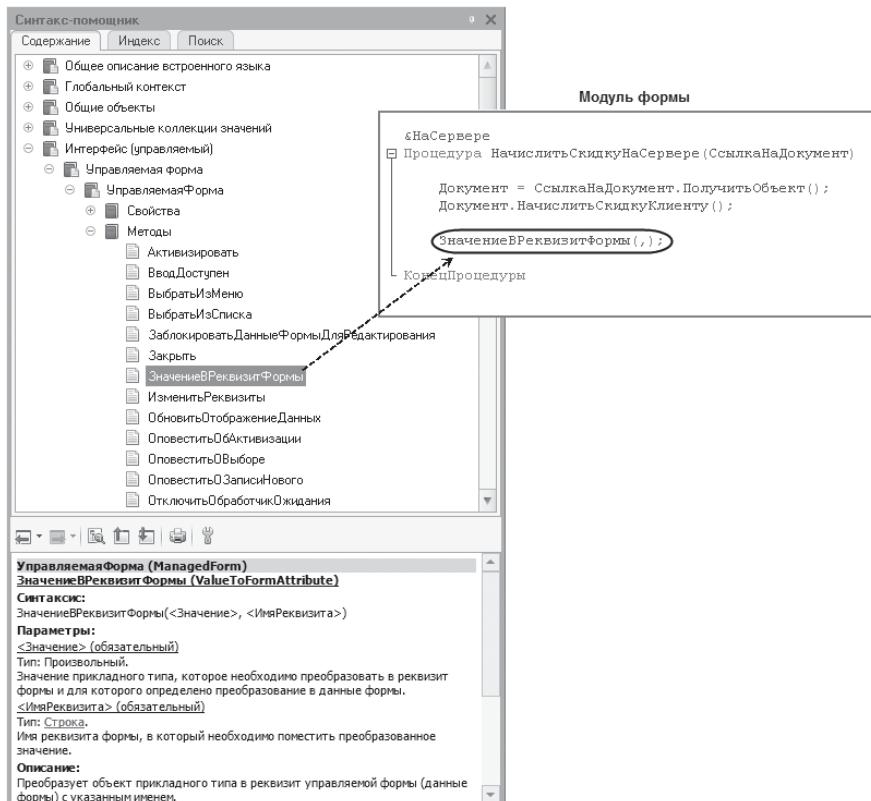


Рис. 81. Перетаскивание свойств, методов из синтакс-помощника

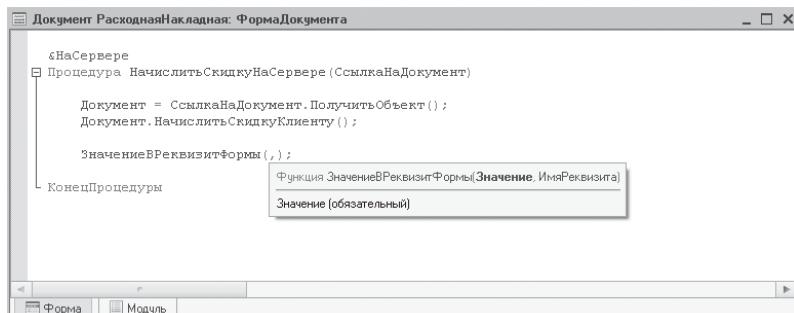


Рис. 82. Вызов контекстной подсказки параметров

Пример 49. Создать копию базы для экспериментов

Предположим, вы разработали конфигурацию, и, чтобы протестировать ее работу, вам нужно на ней «поеэкспериментировать». И, предположим, что ваша конфигурация содержится в файловой информационной базе, которая находится на том же компьютере, что и конфигуратор. Для этого вам нужно создать копию своей базы для экспериментов, чтобы «не портить» исходную конфигурацию.

В случае файловой информационной базы вы можете легко сделать копию исходной базы следующим образом.

Запустите «1С:Предприятие» и в списке информационных баз выделите исходную базу. Под списком баз в окне запуска находится строка, содержащая путь к выделенной базе. Скопируйте этот путь в буфер обмена (рис. 186).

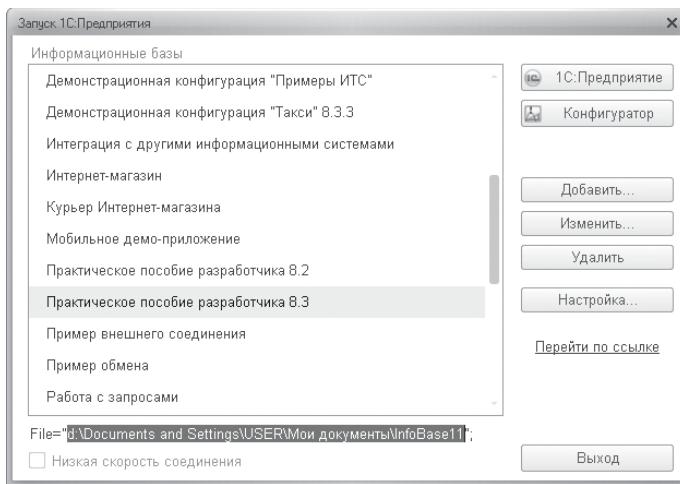


Рис. 83. Окно запуска «1С:Предприятия»

Затем откройте этот каталог в проводнике. Скопируйте из него файл 1Cv8.1CD в буфер обмена (рис. 84).

Поместите копию базы в нужный каталог и подключите этот каталог как существующую информационную базу (рис. 85).

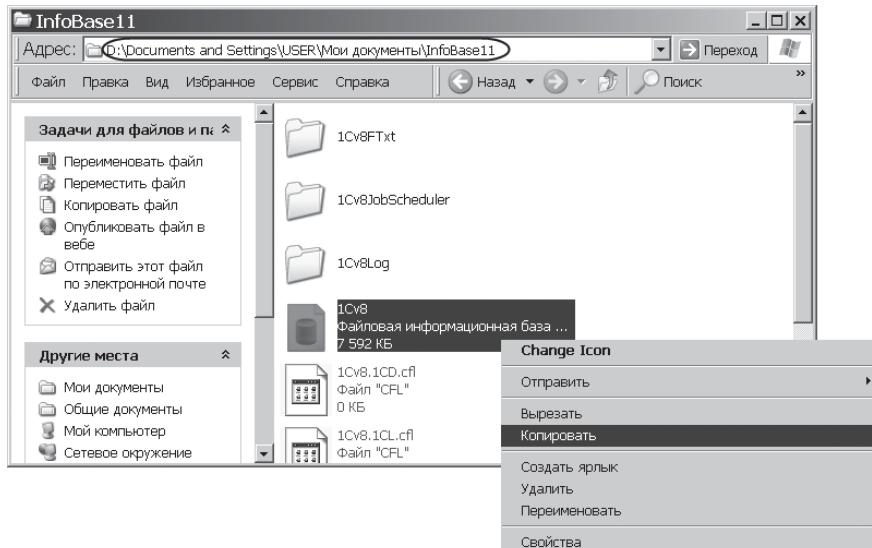


Рис. 84. Копирование файла с исходной информационной базой

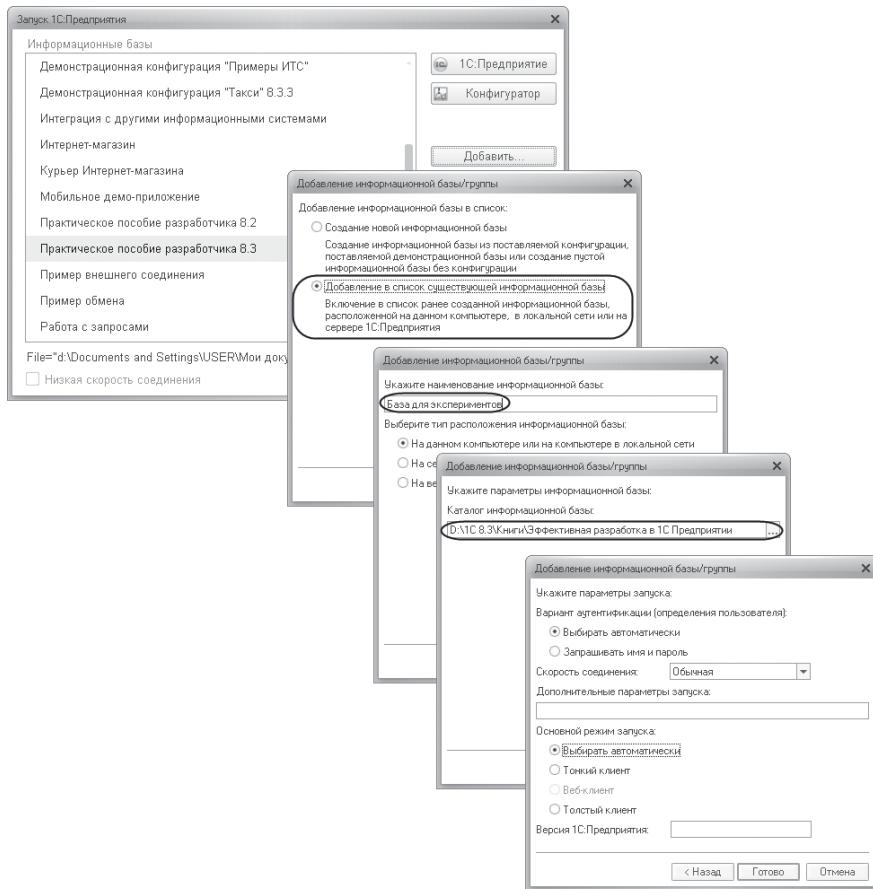


Рис. 85. Добавление в список информационных баз «1С:Предприятия» существующей информационной базы

ПРИМЕЧАНИЕ

Если вы забыли, как это делается, напомним, что в окне запуска «1С:Предприятия» вам нужно нажать кнопку Добавить, затем выбрать опцию Добавление в список существующей информационной базы. Нажмите Далее, укажите наименование информационной базы, на которой вы будете экспериментировать. Нажмите Далее, укажите каталог, в который вы поместили копию исходной базы. Нажмите и в последнем окне нажмите кнопку Готово.

После этого вы можете запустить вашу экспериментальную базу в конфигураторе или в режиме 1С:Предприятие и тренироваться на ней (рис. 86).

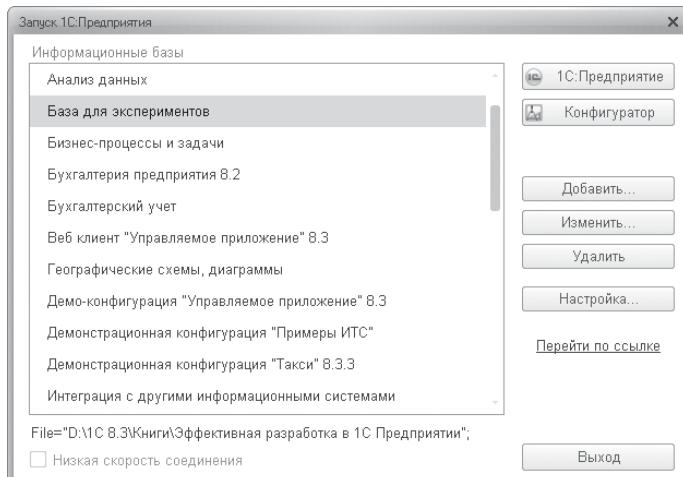


Рис. 86. Окно запуска «1С:Предприятие»

Не забудьте в начале работы с конфигурацией выполнить команду меню Конфигурация > Открыть конфигурацию.

ПРИМЕЧАНИЕ

Чтобы создать копию информационной базы, вы можете также выгрузить исходную базу в файл (Администрирование > Выгрузить информационную базу...), а затем загрузить ее из этого файла в новую пустую конфигурацию. Этот способ более универсальный, так как подходит и для клиент-серверного варианта работы. Но он более медленный. Для файловой базы показанный выше способ – самый эффективный и удобный вариант.

Пользоваться автоматизацией там, где это возможно

Разработка реально работающего прикладного решения в системе «1С:Предприятие» – процесс долгий и трудоемкий. Поэтому конфигуратор содержит многочисленные возможности, позволяющие автоматизировать ваш ручной труд. Речь идет о таких инструментах, как редактор формы, конструктор запроса, контекстная подсказка, шаблоны текста, отладчик, рефакторинг и других.

Все эти механизмы и инструменты полезно знать и вовремя использовать, иначе действительно процесс разработки может стать слишком сложным и трудозатратным.

При разработке форм наиболее просто и удобно сначала создать реквизиты или команды в редакторе формы, а затем мышью перетащить их в окно элементов формы. При этом в дереве элементов формы будут созданы элементы подходящего вида для отображения данных реквизитов, или кнопки для выполнения команд, которые будут автоматически связаны со своими источниками. При удалении реквизитов или команд также автоматически удалятся и связанные с ними элементы формы.

Кроме того, в редакторе формы существует удобный и быстрый способ создания обработчиков событий или команд с помощью контекстного меню элемента формы. При этом будет автоматически установлена связь события или команды и ее процедуры-обработчика. При удалении этой привязки из палитры свойств элемента формы или команды также автоматически будет удалена и сама процедура из модуля формы.

Таким образом, использование вышеперечисленных приемов позволяет сделать разработку форм быстрее и проще, избавляет вас от повторения рутинных действий и помогает сосредоточиться на действительно сложных моментах разработки.

К таким моментам, безусловно, можно отнести написание кода на встроенному языке. Чтобы избежать ошибок, нужно стараться максимально автоматизировать этот процесс и писать вручную как можно меньше. Это позволит вам «написать» программный код не только быстро, но и правильно!

Прежде всего, для этого мы рекомендуем использовать контекстную подсказку. Вы можете вызвать окно с подсказкой в любой произвольный момент при наборе текста модуля, а также окно может появляться автоматически после набора точки, кавычки, знака равенства и т. д.

Чтобы не держать в голове количество и тип параметров, которые надо передать в какой-либо метод объекта, очень полезно использовать контекстную подсказку параметров. Если созданные вами процедуры описаны по определенным правилам, то для них так же, как для функций и процедур встроенного языка, будет появляться контекстная подсказка параметров с описанием их типа и назначения.

Другой эффективный способ сокращения вашего ручного труда – это использование шаблонов текста, как стандартных, так и пользовательских.

Стандартные шаблоны содержат наборы управляющих конструкций, шаблоны для работы с прикладными объектами и другие часто используемые функции глобального контекста.

Пользовательские шаблоны создаются отдельно и подключаются к конфигурации. Вы можете создавать свои или использовать уже созданные кем-то шаблоны текста.

И те, и другие шаблоны вы можете копировать и вставлять или перетаскивать в текст модуля. Наиболее эффективно настроить возможность автоподстановки текста шаблонов в модуль.

Очень важно, чтобы программный код был написан не только правильно, но и понятно. Понятно не только вам, но и другим разработчикам, которые затем, возможно, будут вносить в него изменения.

Чтобы быстро и «красиво» оформить программный код, в конфигураторе существует набор команд для форматирования текстов модулей. Вы можете применять их как к выделенному блоку текста, так и сразу ко всему выделенному модулю, что, безусловно, сокращает ваш ручной труд.

Очень удобно отформатировать выделенный блок текста с использованием синтаксического отступа. Также вы можете добавить/удалить комментарии или символы переноса строки сразу во всем выделенном тексте.

В конфигураторе есть конструктор запроса и конструктор запроса с обработкой результата, которые позволяют значительно облегчить ваш труд при создании и редактировании текста запросов. Использование этих конструкторов позволяет вам быстро написать синтаксически правильный запрос, а также найти и исправить свои ошибки при редактировании текста запроса.

Для создания форматной строки и строк на разных языках удобно использовать соответствующие конструкторы, которые позволят сделать это быстро и главное без ошибок.

В окне редактирования текстов интерфейса вы можете автоматизировать процесс редактирования строк на разных языках. Кроме того, этот механизм вы можете эффективно использовать и для массового переименования полей отчетов, форм, кнопок и т. п. или изменения других совпадающих значений у объектов конфигурации, даже если в конфигурации определен только один язык.

В данном разделе речь также пойдет о наиболее полезных и удобных приемах использования отладчика для устранения ошибок. Вы узнаете, как посмотреть значения переменных в момент остановки приложения, как проследить изменение переменных по шагам. Как остановить исполнение перед возникновением ошибки, как узнать, откуда была вызвана процедура с ошибкой. Как остановить исполнение при наступлении некоторого условия, как посмотреть результат выполнения запроса с помощью отладчика и т. д.

Кроме того, мы расскажем, как запустить отладочный сеанс из конфигуратора от имени другого пользователя, как подключить отладчик к работающему сеансу, и рассмотрим общую наиболее эффективную стратегию исправления ошибок.

Чтобы поддерживать и модифицировать прикладное решение, в коде конфигурации периодически нужно наводить «порядок». Этот процесс называют рефакторингом. С помощью средств рефакторинга вы можете автоматически создать описание процедуры, выделить часть кода в отдельную процедуру, автоматически изменить имя переменной или процедуры. А также можете создать обработку оповещения для использования немодальных методов.

Таким образом, рефакторинг не меняет внешнее поведение прикладного решения, но делает код конфигурации более читаемым, понятным и легким в сопровождении.

Перечисленные возможности показаны ниже на небольших коротких примерах.

Пример 50. Создать поле ввода

Например, вам нужно создать поле, в котором пользователь мог бы выбирать элементы справочника Клиенты. Это будет ссылочное поле, в котором будут отображаться ссылки на элементы справочника.

Самый простой и удобный способ следующий: в окне реквизитов формы добавьте реквизит Клиент типа СправочникСсылка.Клиенты и перетащите его в окно элементов формы. В дереве элементов формы автоматически будет создано поле вида Поле ввода, а в свойстве ПутьКДанным у него будет указана ссылка на соответствующий реквизит.

Результат работы вы сразу же можете увидеть в нижней части окна редактора формы. Причем при выделении элемента в дереве элементов формы он сразу же выделяется в окне предварительного просмотра, и наоборот (рис. 87).

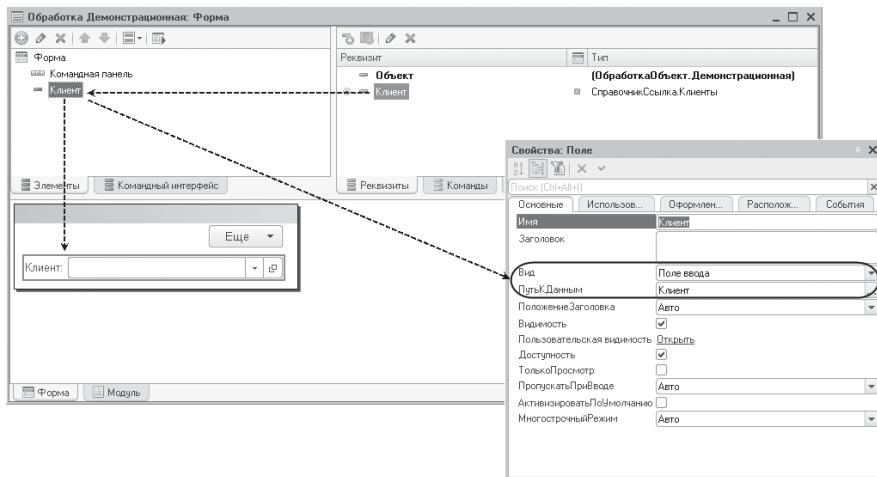


Рис. 87. Создание реквизита формы и связанного с ним поля ввода

Аналогичным образом, если нужно поле для ввода и редактирования числа, добавьте реквизит типа Число (для редактирования строки – строковой реквизит и т. п.) и перетащите его в форму.

Пример 51. Создать поле табличного документа

Чтобы создать поле табличного документа, в окне реквизитов формы добавьте реквизит типа ТабличныйДокумент и перетащите его в окно элементов формы.

В дереве элементов формы автоматически будет создано поле вида Поле табличного документа, а в свойстве ПутьКДанным у него будет указана ссылка на соответствующий реквизит. Также заметьте, что при наведении мыши на элемент формы всплывает подсказка, которая отображает реквизит, связанный с этим элементом (рис. 88).

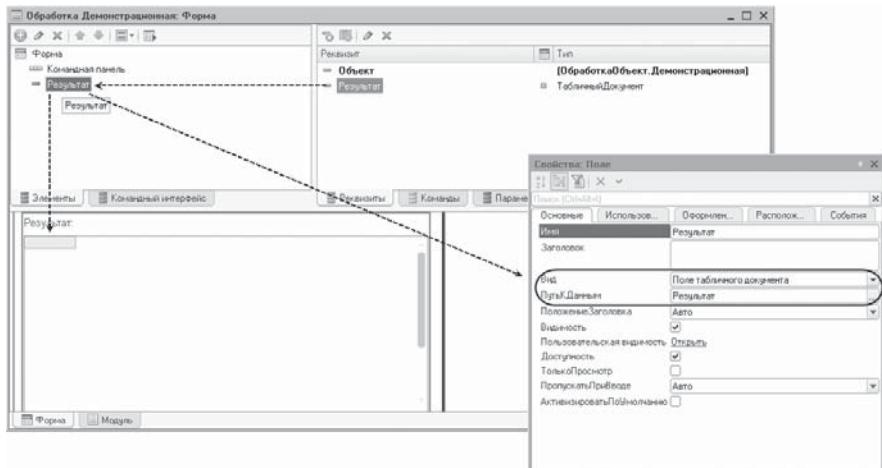


Рис. 88. Создание реквизита формы и связанного с ним поля табличного документа

При выборе типа реквизита (при нажатии кнопки выбора) для облегчения поиска нужного типа вы можете воспользоваться строкой поиска в окне выбора типа. В строке поиска окна редактирования типа данных введите символы, которые требуется найти в имени типа (рис. 89).

Аналогичным образом вы можете создать в форме поле текстового документа и поле форматированного документа. При этом реквизиты должны иметь соответствующий тип – ТекстовыйДокумент или ФорматированыйДокумент.

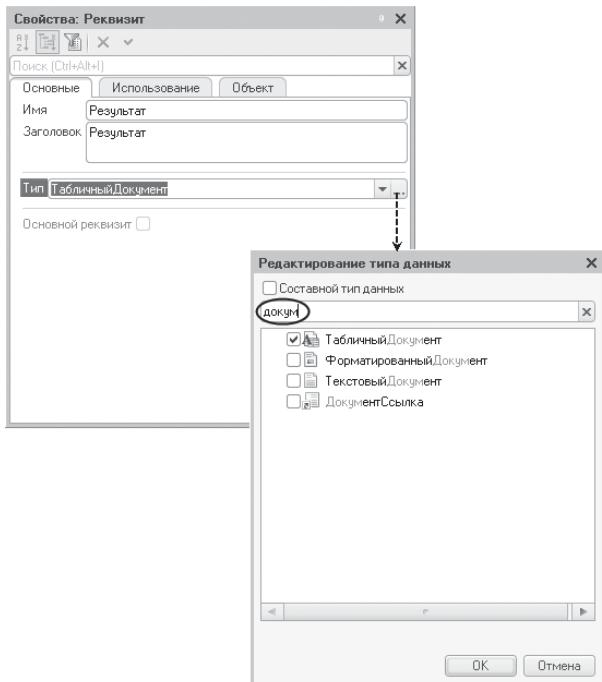


Рис. 89. Поиск типов объектов в строке поиска при выборе типа

Пример 52. Создать поле флажка

Чтобы создать поле флажка, в окне реквизитов формы добавьте реквизит типа Булево и перетащите его в окно элементов формы.

В дереве элементов формы автоматически будет создано поле вида Поле флажка, а в свойстве ПутьКДанным у него будет указана ссылка на соответствующий реквизит (рис. 90).

ПРИМЕЧАНИЕ

Для создания поля флажка вы можете также добавить в форму числовой реквизит. И после перетаскивания его в дерево элементов формы установить у получившегося поля свойство Вид как Поле флажка. При этом если флажок снят, то значение реквизита равно нулю, если флажок установлен, то значение реквизита равно единице.

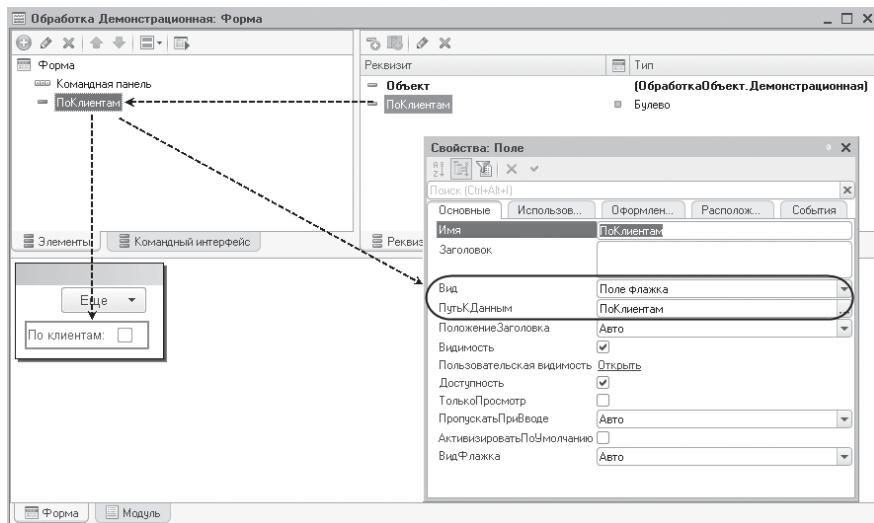


Рис. 90. Создание реквизита формы и связанного с ним поля флажка

Пример 53. Создать поле картинки

Чтобы создать поле картинки, в окне реквизитов формы добавьте реквизит типа Картинка и перетащите его в окно элементов формы.

В дереве элементов формы будет создано поле вида Поле картинки, у которого в свойстве ПутьКДанным будет указана ссылка на соответствующий реквизит (рис. 91).

Чтобы картинка отображалась в форме, саму картинку вам нужно программно поместить в соответствующий реквизит формы, например в обработчике события ПриСозданииНаСервере (листинг 1).

Листинг 1. Обработчик события формы «ПриСозданииНаСервере»

```
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    Картинка = БиблиотекаКартинок.Телефон;
КонецПроцедуры
```

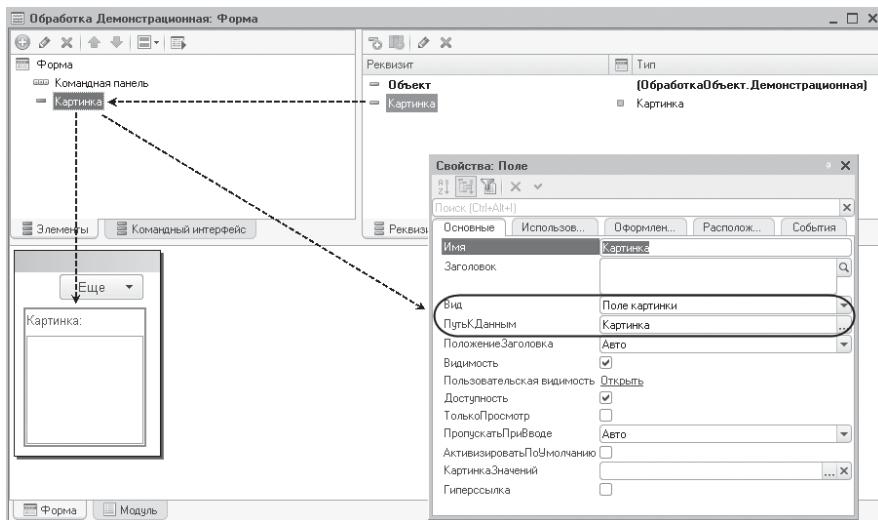


Рис. 91. Создание реквизита формы и связанного с ним поля картинки

ПРИМЕЧАНИЕ

Картинки, хранимые в данных объектов, вы также можете показывать в форме. Для этого вы можете использовать строковый реквизит объекта, в котором будет храниться навигационная ссылка на данные картинки. При этом, естественно, у вас должен быть еще какой-то программный код, который помещает картинку в виде двоичных данных в нужный реквизит объекта. А в обработчике события ПриЧтенииНаСервере (или тогда, когда это нужно) программный код, который получает навигационную ссылку на эту картинку и помещает ее в реквизит формы.

Пример 54. Создать поле переключателя

Чтобы создать поле переключателя, в окне реквизитов формы добавьте реквизит типа Число или Стока и перетащите его в окно элементов формы.

После этого в дереве элементов формы будет создано поле вида Поле ввода, у которого в свойстве ПутьКДанным будет указана ссылка на соответствующий реквизит.

Особенность заключается в том, что здесь вас не устраивает вид поля, автоматически предложенный платформой. Измените свойство Вид этого поля на Поле переключателя (рис. 92).

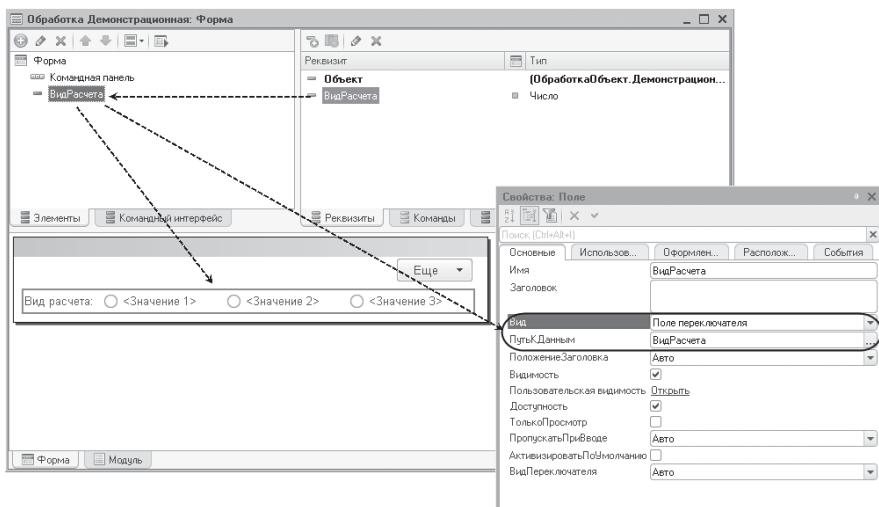


Рис. 92. Создание реквизита формы и связанного с ним поля переключателя

Чтобы значения переключателя отображались в форме, вам нужно программно или интерактивно в конфигураторе задать свойство СписокВыбора для этого переключателя.

Аналогичным образом вы можете создать в форме поле индикатора и поле полосы регулирования. При этом вид поля должен иметь соответствующее значение – Поле индикатора или Поле полосы регулирования.

Пример 55. Создать поле HTML-документа

Чтобы создать поле HTML-документа в окне реквизитов формы, добавьте реквизит типа Стока и перетащите его в окно элементов формы.

В дереве элементов формы будет создано поле, у которого в свойстве ПутьКДанным будет указана ссылка на соответствующий реквизит. После этого установите свойство Вид этого поля как Поле HTML документа (рис. 93).

В строковом реквизите (в нашем примере Содержимое) будет храниться адрес для доступа к веб-контенту (URL интернет-ресурса) или сам HTML-код, и эти данные будут отображаться в форме в поле HTML-документа.

Аналогичным образом вы можете создать в форме поле текстового документа (вид поля – Поле текстового документа).

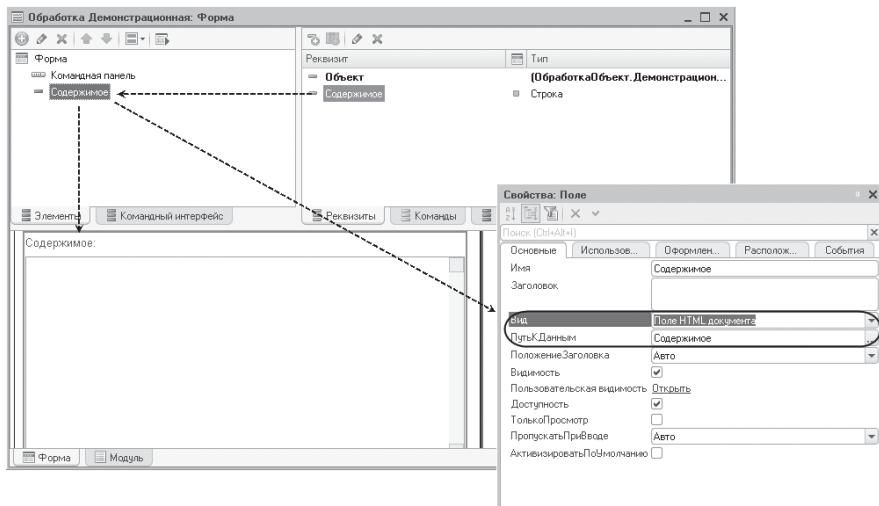


Рис. 93. Создание реквизита формы и связанного с ним поля HTML-документа

Пример 56. Удалить элемент формы или команду

Как правило, если элемент формы не нужен, то не нужны и отображаемые в нем данные (реквизиты), за редким исключением. То же самое относится и к командам.

Если вы будете удалять отдельно элементы и связанные с ними реквизиты формы, то мало того, что это долго и неудобно. Если в процессе удаления вы отвлеклись, может получиться так, что данные в форме есть, а отображаться им негде (тогда зачем получаются эти данные?). Или наоборот – элементы в форме есть, а данных для отображения в них нет (тогда эти элементы все равно не будут показаны в форме).

Поэтому реквизиты и элементы формы наиболее эффективно удалять «вопарно».

Для удаления одновременно и реквизита, и связанного с ним элемента формы выделите реквизит на закладке Реквизиты и нажмите кнопку Удалить в командной панели окна реквизитов. На вопрос конфигуратора об удалении связанных элементов ответьте утвердительно (рис. 94).

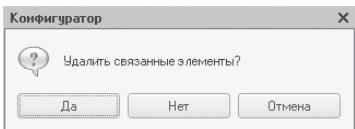


Рис. 94. Вопрос об удалении связанных элементов

Соответствующие элементы сразу же удалятся и из дерева элементов формы, и из окна предварительного просмотра формы.

Удаление команды и связанной с ней кнопки выполняется точно так же. С той лишь разницей, что команду вам нужно выбрать на закладке Команды.

ПРИМЕЧАНИЕ

Для удаления реквизита вы можете также выделить его и нажать кнопку Del или выполнить пункт Удалить из контекстного меню реквизита.

Пример 57. Создать кнопку

При разработке формы недостаточно создать команды формы и кнопки для их выполнения. Дело в том, что, пока не установлена связь между ними, никакие команды выполняться не будут. Эта связь задается в свойстве кнопок/гиперссылок ИмяКоманды.

Создавать отдельно команды и кнопки/гиперссылки формы и вручную устанавливать связь между ними – долго и неэффективно.

Самый простой и удобный способ для этого – сначала создать команду формы и затем мышью перетащить ее в окно элементов формы. При этом в форме будет создана кнопка, связанная с этой командой в свойстве элемента ИмяКоманды. После этого можно сразу перейти в процедуру обработчика и написать нужный вам алгоритм.

Рассмотрим это на примере. Предположим, вам нужно поместить в командную панель формы кнопку Рассчитать, при нажатии на которую будет производиться некоторый перерасчет данных.

Для этого в правом верхнем окне редактора формы на закладке Команды добавьте команду формы Рассчитать и перетащите ее в дерево элементов формы, но не на пустое место, а в командную панель. Заметьте, что при этом будет создана не просто кнопка, а кнопка командной панели (свойство Вид), и в свойстве ИмяКоманды будет указана ссылка на команду Рассчитать (рис. 95).

Если теперь в дереве элементов формы навести мышь на эту кнопку, то всплывет подсказка, которая отображает связанную с кнопкой команду.

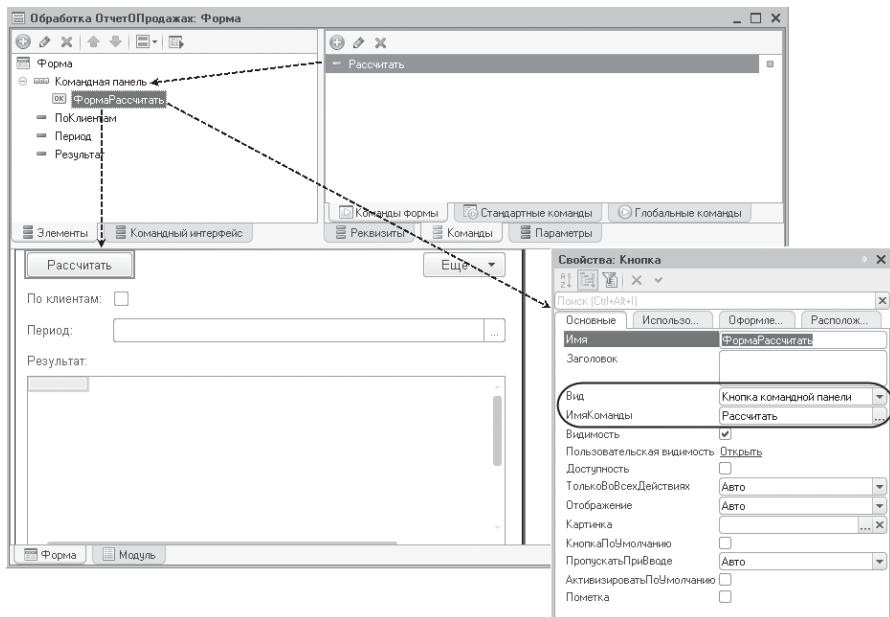


Рис. 95. Конструирование произвольной формы в редакторе формы

Теперь вам нужно создать обработчик команды, которая будет выполняться при нажатии на кнопку командной панели формы с именем ФормаРассчитать.

Для этого вызовите контекстное меню кнопки ФормаРассчитать в дереве элементов формы (рис. 96).

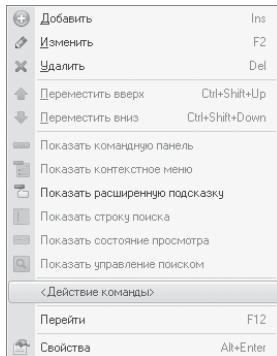


Рис. 96. Создание обработчика команды из контекстного меню кнопки

Выполните пункт <Действие команды>. На вопрос конфигуратора о типе обработчика команды ответьте, что вы хотите создать клиентский обработчик команды формы с вызовом из него процедуры, выполняющейся на сервере (рис. 97).

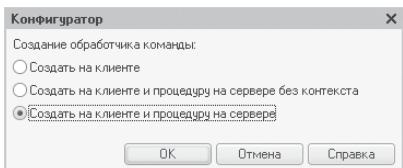


Рис. 97. Выбор типа обработчика команды формы

Дело в том, что обработчик команды всегда выполняется на клиенте, так как нажатие на кнопку – это интерактивное действие пользователя. А пересчет данных обычно выполняется на сервере. При этом в данном случае будет использоваться контекст формы. Если контекст формы для пересчета не нужен, лучше выбрать второй вариант ответа («Создать на клиенте и процедуре на сервере без контекста»).

После выбора типа обработчика команды в модуле формы будут созданы шаблоны двух процедур: клиентской процедуры Рассчитать() и серверной процедуры РассчитатьНаСервере(), которая вызывается из процедуры Рассчитать().

В результате модуль формы обработки будет выглядеть следующим образом (листинг 2).

Листинг 2. Модуль формы

```
&НаКлиенте
Процедура Рассчитать(Команда)
    РассчитатьНаСервере();
КонецПроцедуры

&НаСервере
Процедура РассчитатьНаСервере()
    // Вставить содержимое обработчика.
КонецПроцедуры
```

После того как вы создали обработчик команды, связанный с кнопкой Рассчитать, вместо пункта <Действие команды>, который означает, что обработчик команды еще не создан, теперь в контекстном меню кнопки вы увидите наименование процедуры-обработчика Рассчитать(). Выполнив этот пункт меню, вы можете в любой момент сразу же перейти к этой процедуре в модуле формы (рис. 98).

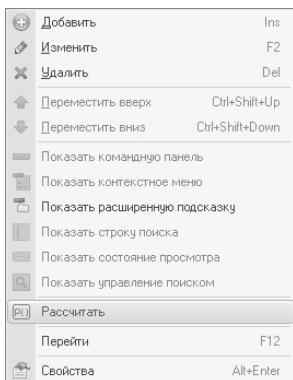


Рис. 98. Переход к обработчику команды из контекстного меню связанной с ней кнопки

При этом в палитре свойств команды, связанной с кнопкой, в свойстве Действие будет автоматически установлена связь команды и ее обработчика (рис. 99).

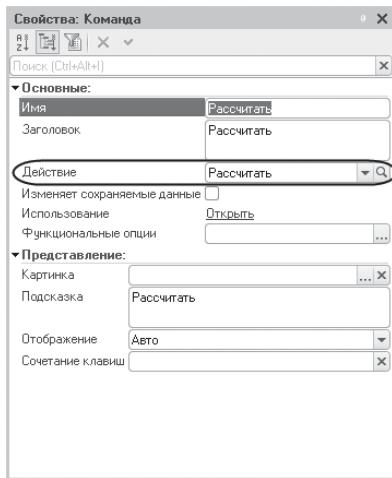


Рис. 99. Установка обработчика команды

Таким образом, вы завершили логическую цепочку действий. Сначала создали команду и перетащили ее на форму – получилась кнопка, связанная с этой командой. Затем из контекстного меню кнопки создали обработчик команды, который будет выполняться при нажатии на эту кнопку.

Пример 58. Создать обработчик события

Обработчики событий бывают двух типов – стандартные и назначаемые.

Стандартные обработчики имеют фиксированные имена и используются в модулях приложения, в модулях объектов и т. п. – например, обработчик события документа ОбработкаПроведения имеет такое же имя в модуле объекта.

Назначаемые обработчики могут иметь произвольные имена, хотя на практике они часто имеют имена, частично совпадающие с именем события. Этот тип обработчиков используется в формах.

Особенность их в том, что наличие в модуле формы процедуры с именем события, например ПриОткрытии(), само по себе еще не гарантирует, что при наступлении события формы ПриОткрытии будет выполнена одноименная процедура из модуля формы. Для этого в свойствах формы эту процедуру нужно в явном виде назначать как процедуру-обработчик соответствующего события.

Вручную создавать процедуру-обработчик события в модуле формы и устанавливать его связь с событием неэффективно.

Выполнить эту привязку можно быстро и легко, всего за один шаг. Для этого существуют два удобных способа – с помощью контекстного меню элемента формы (или самой формы) и с помощью палитры свойств. В большинстве случаев логичнее и удобнее использовать контекстное меню, но в некоторых ситуациях для этого более удобна палитра свойств. Рассмотрим оба способа подробнее.

С помощью контекстного меню. Предположим, вам нужно создать обработчик события ПриИзменении, которое будет возникать при изменении поля ввода Период.

Для этого вызовите контекстное меню элемента формы Период и откройте подменю События. В нем вы увидите список событий, возникающих у выделенного элемента формы. Все они заключены в угловые скобки – это значит, что для элемента Период еще не создано ни одного обработчика события (рис. 100).

Выполните пункт с нужным вам событием <ПриИзменении>. На вопрос конфигуратора о типе обработчика команды ответьте, что вы хотите создать клиентский обработчик события, так как это событие происходит на клиенте.

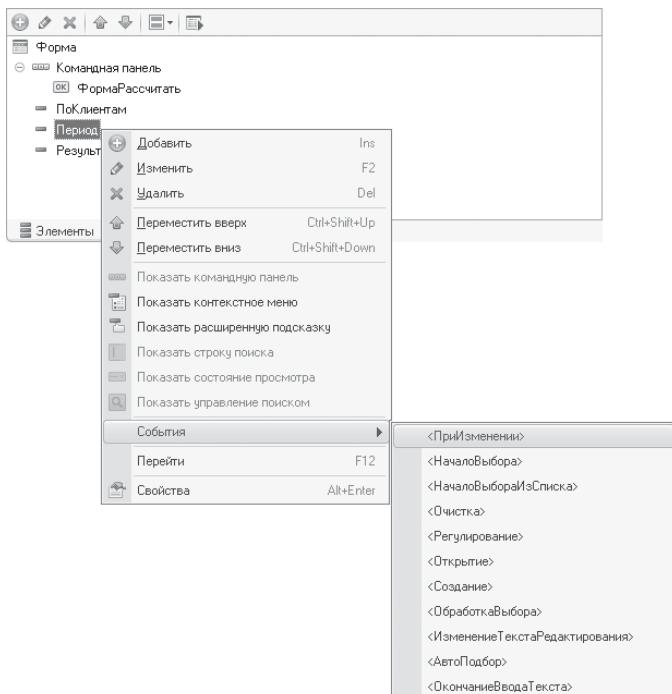


Рис. 100. Создание обработчика события из контекстного меню элемента формы

После этого в модуле формы будет создан шаблон клиентской процедуры-обработчика, в который нужно поместить код, выполняющийся при изменении поля ввода Период (листинг 3).

Листинг 3. Модуль формы

```
&НаКлиенте
Процедура ПериодПриИзменении(Элемент)
    // Вставить содержимое обработчика.
КонецПроцедуры
```

Теперь, если вы вызовите контекстное меню элемента формы Период, то в подменю События вы увидите наименование процедуры-обработчика ПериодПриИзменении(), к которой вы можете в любое время перейти, выполнив этот пункт меню (рис. 101).

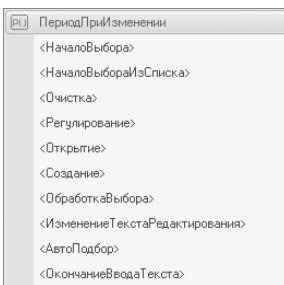


Рис. 101. Переход к обработчику события из контекстного меню элемента формы

Обратите внимание, что у существующего и назначенного обработчика исчезли угловые скобки и появилась пиктограмма слева от наименования процедуры – Р().

При этом в палитре свойств поля ввода Период для события ПриИзменении будет назначен обработчик – процедура ПериодПриИзменении() (рис. 102).

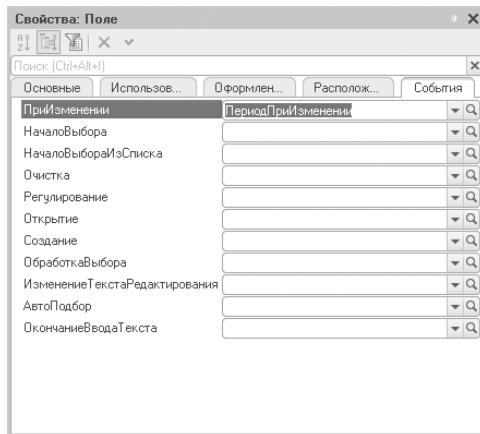


Рис. 102. Установка обработчика события

ПРИМЕЧАНИЕ

Если вам нужно создать обработчик события самой формы, а не ее элемента, то вы можете выделить корень дерева элементов формы и вызвать его контекстное меню. Дальше – по аналогии с вышеуказанным.

С помощью палитры свойств. Другой способ создания обработчика события – с помощью палитры свойств. Возможна такая ситуация, когда вы открыли палитру свойств элемента формы, установили какие-то свойства, затем открыли закладку События, чтобы перейти к обработчику какого-то события. И увидели, что обработчик нужного вам события по каким-то причинам не назначен (возможно, вас отвлекли, и вы забыли об этом).

В этом случае удобнее создать обработчик события сразу же из палитры свойств, нажав кнопку открытия справа от имени обработчика события (рис. 103).

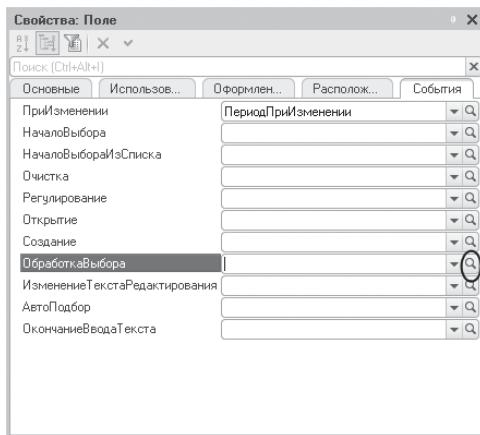


Рис. 103. Создание обработчика события из палитры свойств элемента формы

После выбора типа обработчика события в модуле формы будет создан шаблон клиентской процедуры-обработчика. Таким образом, вы получите такой же результат, что и в предыдущем примере, при этом будет установлена связь события и процедуры-обработчика в палитре свойств элемента формы.

Пример 59. Удалить обработчик события

Если обработчик события для элемента формы больше не нужен, то вам нужно удалить и саму процедуру обработчика из модуля формы, и привязку к ней соответствующего события в палитре свойств элемента формы.

Делать это по отдельности долго и неудобно. Кроме того, это может привести к ошибкам.

Например, если вы удалили процедуру-обработчик в модуле формы и забыли удалить привязку к ней в палитре свойств, вы получите ошибку при синтаксической проверке модуля. Если же вы удалили привязку к обработчику, в этот момент вас отвлекли, и в модуле осталась процедура обработчика, то ошибки не будет, но в модуле формы останется ненужный «мусор», что тоже нехорошо.

Поэтому наиболее эффективно удалять одновременно и привязку события к процедуре-обработчику в палитре свойств, и саму процедуру из модуля формы.

Для этого выделите процедуру-обработчик в палитре свойств элемента формы и нажмите клавишу Del и затем Enter. После этого конфигуратор предложит удалить эту процедуру из модуля формы (рис. 104).

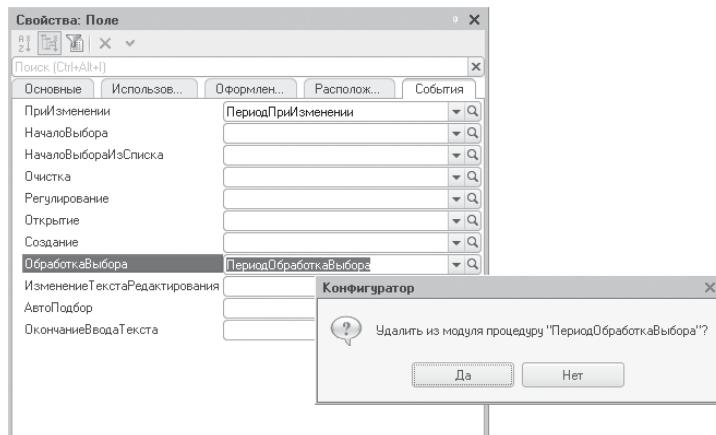


Рис. 104. Установка и удаление обработчика события

Согласившись с этим, вы также быстро, за один шаг, удаляете и привязку обработчика к событию в палитре свойств, и саму процедуру из модуля формы.

Аналогичным образом вы можете удалить обработчик команды, очистив свойство Действие в палитре свойств команды.

Советуем делать именно так, чтобы не оставалось неиспользуемых процедур в тексте модуля формы («отвязанных» в палитре свойств, но присутствующих в модуле процедур-обработчиков).

Также если вы копируете какую-то процедуру из формы в форму, не забывайте назначать ее в качестве обработчиков событий или команд, если она предназначалась для этих целей. Иначе эти процедуры также «повиснут в воздухе» и не будут выполняться в тот момент, когда вы этого ожидаете.

Если все же некоторые процедуры у вас «повисли в воздухе», то проверить наличие неиспользуемых процедур в модулях вы можете, выполнив проверку конфигурации (Конфигурация > Проверка конфигурации).

Пример 60. Подсказать пользователю назначение элемента

Платформа «1С:Предприятие» позволяет выводить подсказку при работе пользователя с различными формами. Отображение подсказок во многих случаях может избавить пользователя от чтения инструкций. Кроме того, сам процесс работы становится более легким и «живым».

Есть много разных способов, как это сделать, но наиболее удобными являются два из них. Это подсказка ввода и расширенная подсказка.

Если вам нужно просто подсказать пользователю, что ему вводить в то или иное поле формы, то удобнее всего воспользоваться для этого поля подсказкой ввода. Если же вам требуется показать более содержательную или отформатированную подсказку (и не только для полей, но и для кнопок, таблиц и др.), то вы можете использовать для этих элементов формы расширенную подсказку.

Рассмотрим подробнее отображение этих подсказок в форме обработки, в которой производится перерасчет данных за период.

Подсказка ввода. Чтобы при заполнении различных полей ввода подсказать пользователю, что вводить в конкретное поле, просто заполните у этого поля ввода свойство ПодсказкаВвода.

Предположим, вы хотите напомнить пользователю, что в поле Период ему нужно выбрать стандартный отчетный период.

Для этого в редакторе формы обработки в конфигураторе откройте палитру свойств поля Период и в свойстве ПодсказкаВвода введите поясняющую строку, например, «Выберите отчетный период» (рис. 105).

В результате при запуске обработки в режиме 1С:Предприятие в поле Период для пользователя будет отображена поясняющая подсказка (рис. 106).

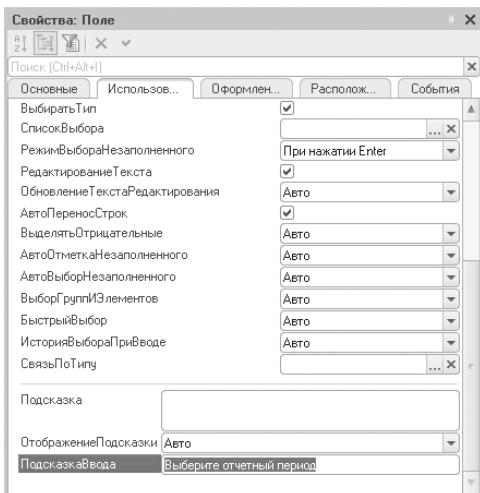


Рис. 105. Установка свойства «ПодсказкаВвода» для поля ввода

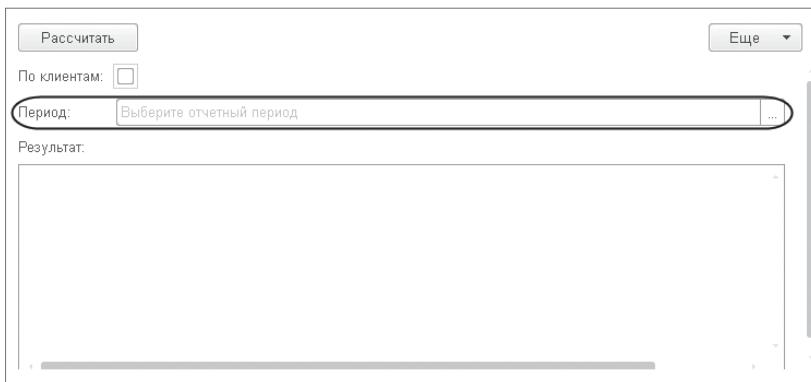


Рис. 106. Подсказка ввода у поля формы

Расширенная подсказка. Продолжим наш пример. Предположим, что у флажка По клиентам в форме обработки вы хотите вывести кнопку (в виде знака вопроса), при нажатии на которую пользователю будет показана расширенная подсказка, объясняющая назначение этого флагка.

Для этого сначала в палитре свойств флажка ПоКлиентам установите свойство ОтображениеПодсказки в значение Кнопка (рис. 107).

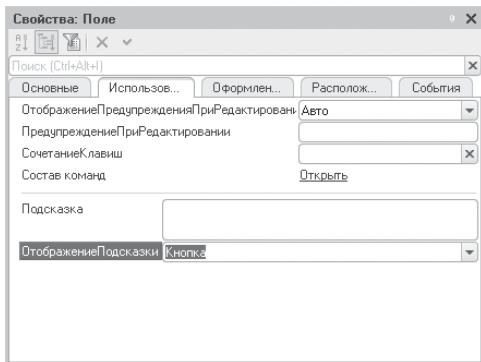


Рис. 107. Установка свойства «ОтображениеПодсказки» для поля флажка

Затем вызовите контекстное меню этого поля в дереве элементов формы и выполните пункт Показать расширенную подсказку. В результате у поля флажка появится подчиненный элемент (вида декорации) Расширенная подсказка (рис. 108).

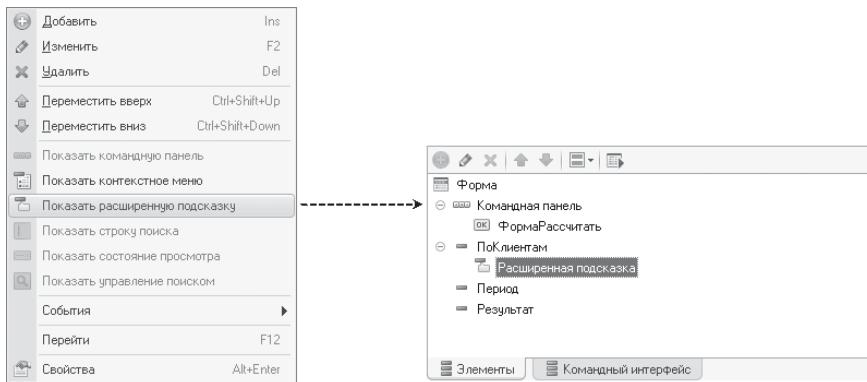


Рис. 108. Отображение расширенной подсказки в дереве элементов формы

В палитре свойств декорации Расширенная подсказка нажмите на кнопку открытия (со значком лупы) у свойства Заголовок. Откроется окно редактирования текста подсказки, в котором вы можете вводить обычный текст, а можете отформатировать его для большей наглядности. Для этого отметьте признак Форматированная строка. Теперь вы можете выделить какие-то слова подсказки цветом, курсивом и т.п. Можете также вставить рисунок или гиперссылку прямо в текст подсказки (рис. 109).

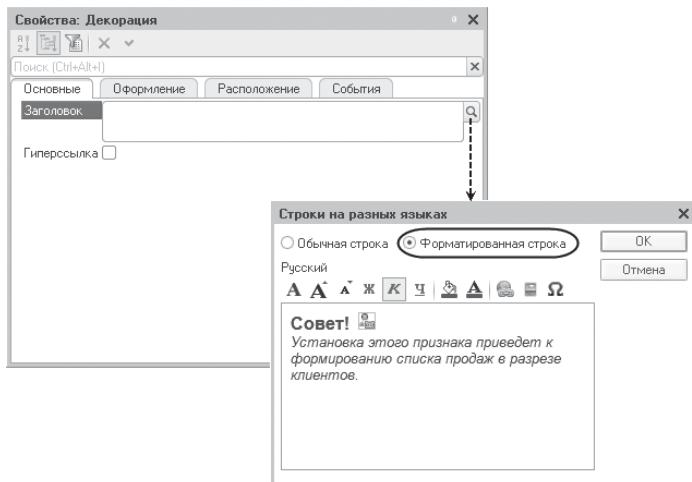


Рис. 109. Редактирование текста расширенной подсказки

Нажмите OK. Для повторного редактирования текста расширенной подсказки вам нужно опять нажать кнопку открытия у свойства Заголовок. Ранее установленное форматирование будет сохранено.

В результате в пользовательском режиме в форме обработки рядом с флажком По клиентам будет отображен знак вопроса, при нажатии на который откроется подробная красиво оформленная подсказка, поясняющая назначение этого флажка (рис. 110).

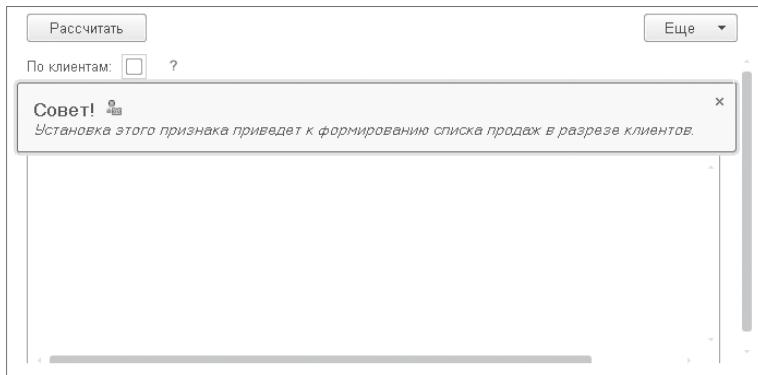


Рис. 110. Отображение расширенной подсказки у флашка

Пример 61. Не писать «вручную» имена переменных, свойств и методов

Все, что входит в контекст какого-либо модуля, становится доступно через контекстную подсказку.

Предположим, в процедуре модуля формы вы ввели локальную переменную АдресКлиента и присвоили ей какое-то значение. Затем вам понадобилось использовать эту переменную еще раз.

Если вы будете снова писать имя переменной вручную, то можете ошибиться (особенно если имя длинное), кроме того это может быть медленно и долго.

Наиболее эффективно в данной ситуации использовать контекстную подсказку. Для этого наберите первые несколько символов имени этой переменной (например, «адр») и нажмите **Ctrl + <Пробел>**. При этом список контекстной подсказки будет позиционирован на найденном соответственно. Поскольку в примере такое соответствие одно, то имя переменной будет сразу же подставлено в текст модуля (рис. 111).

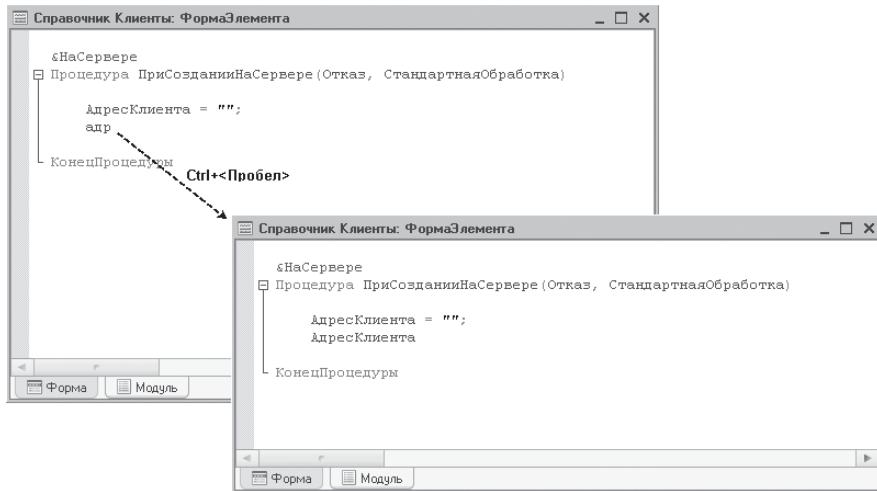


Рис. 111. Использование контекстной подсказки при написании имени переменной

Эта рекомендация, не писать «вручную», справедлива также для свойств и методов модуля.

Предположим, вам нужно программно установить заголовок формы. Для этого вам нужно использовать в модуле формы ее свойство Заголовок.

Чтобы написать имя свойства правильно и быстро, в модуле формы наберите «Заго» и нажмите **Ctrl + <Пробел>**. Имя свойства сразу же подставится в модуль.

Или, например, вам нужно в модуле объекта справочника вызвать его метод **Модифицированность()**.

Для этого в модуле объекта напишите «**мод**» и нажмите **Ctrl + <Пробел>**. При этом откроется список контекстной подсказки, позиционированный на первом найденном соответствии (**МодельСодержимогоXS**). Если вы продолжите набор при открытом списке контекстной подсказки и введете символ «**и**», то он позиционируется на следующем соответствии с набранным текстом, в данном случае – на методе **Модифицированность** (рис. 112).

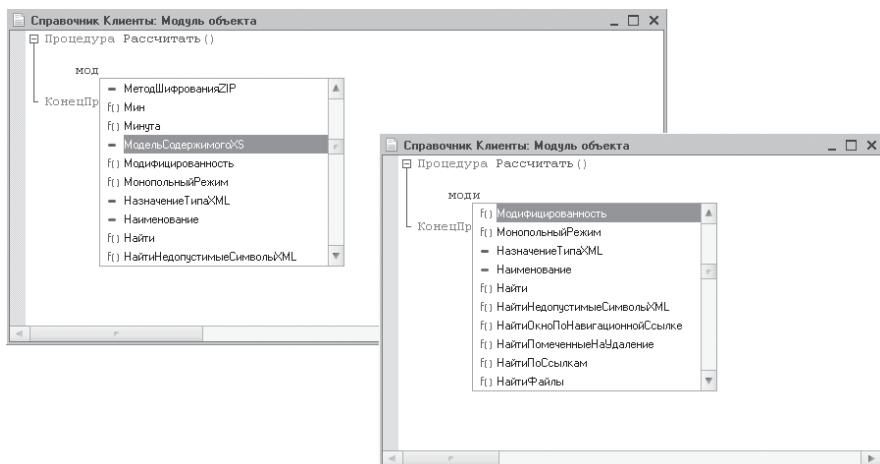


Рис. 112. Использование контекстной подсказки при написании имени метода объекта

При нажатии клавиши **Enter** выбранный вами метод **Модифицированность()** будет подставлен в текст модуля.

Если же вдруг вы, например, установили курсор в другое место модуля, и список контекстной подсказки закрылся, то, чтобы вновь его открыть, нужно вернуться в прежнюю позицию и снова нажать **Ctrl + <Пробел>**.

Пример 62. Настройте контекстную подсказку

Помимо вызова контекстной подсказки после набора произвольного фрагмента текста нажатием **Ctrl + <Пробел>**, конфигуратор выдает автоматическую подсказку после набора знака равенства/неравенства, точки, кавычки, запятой и открывающей скобки. Это стандартная возможность системы, которую вы можете посмотреть и настроить в окне параметров конфигуратора: **Сервис > Параметры > Модули > Контекстная подсказка** (рис. 113).

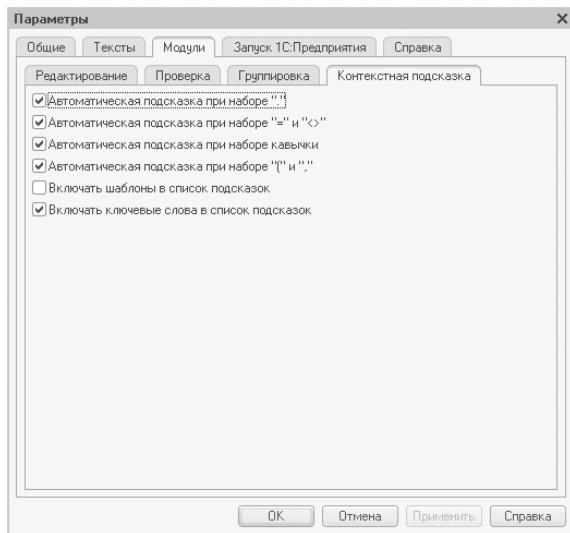


Рис. 113. Настройка параметров конфигуратора

Пример 63. Подсказка после знака равенства

Предположим, вам нужно установить свойство формы **ТолькоПросмотр** в значение **Истина**.

Наберите имя свойства с помощью контекстной подсказки («**тол**», **Ctrl + <Пробел>** и **Enter**). После этого наберите на клавиатуре пробел и знак равенства **<=>**.

Если автоматическая контекстная подсказка при наборе знака равенства включена, то после небольшого ожидания вы увидите список контекстной подсказки, уже позиционированный на значении **Истина** (рис. 114).

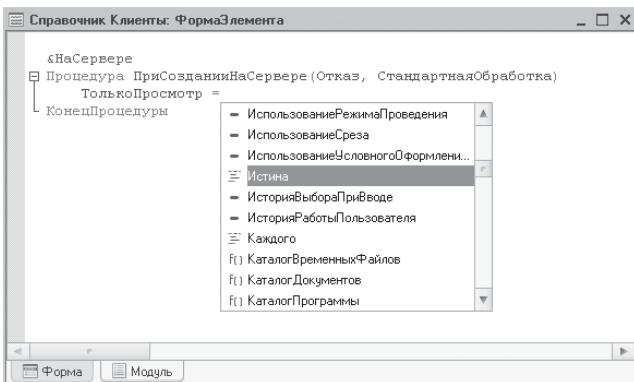


Рис. 114. Список контекстной подсказки

Так произошло потому, что система «поняла», что выражение слева от знака равенства булевого типа. Если вам требуется выбрать другое значение (например, Ложь), то нужно набрать на клавиатуре первые несколько символов этого слова. При этом список будет последовательно позиционироваться на строках, совпадающих с набираемым текстом.

Если слева от знака равенства находится переменная, значение которой имеет тип системного перечисления, то после набора знака равенства/неравенства» в списке контекстной подсказки вам будет предложено перечисление соответствующего типа, а затем, через точку, значения этого перечисления.

Пример 64. Подсказка после точки

Точка является зарезервированным символом «1С:Предприятия» и указывает системе на то, что слева от нее находится объект. Система, если это возможно, определяет тип этого объекта и предлагает для выбора свойства и методы объекта, доступные в контексте того модуля, откуда была вызвана контекстная подсказка.

Рассмотрим пример. Предположим, при создании формы объекта вам нужно запомнить значение реквизита объекта в какой-то переменной.

Чтобы получить данные объекта из формы, вам потребуется обратиться к основному реквизиту формы Объект, который содержит все данные объекта. В процедуре модуля формы напишите идентификатор Объект и после него поставьте точку «.».

Если автоматическая контекстная подсказка при наборе точки включена, то после этого откроется список контекстной подсказки, в котором вы сможете

выбрать нужный реквизит или свойство объекта, доступные в форме (рис. 115).

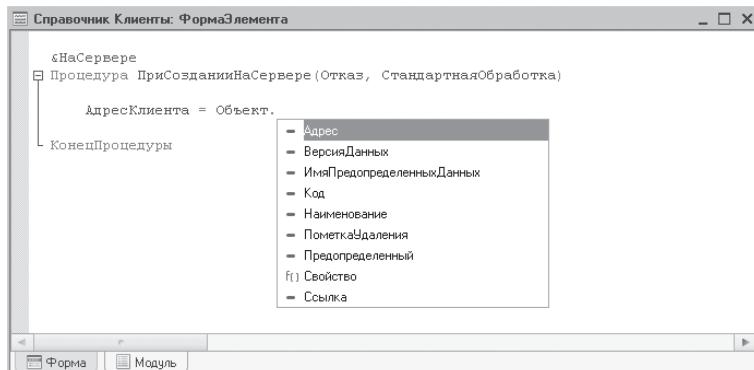


Рис. 115. Использование контекстной подсказки при написании имени реквизита объекта

По аналогии с вышеизложенным, если локальной переменной присвоить данные какого-то объекта, при наборе точки после имени этой переменной автоматически вызывается контекстная подсказка, содержащая список свойств и методов, доступных объекту, данных которого она содержит.

Хочется все же предостеречь от бездумного использования контекстной подсказки при наборе точки. Желательно четко понимать, что вы хотите сделать, – например, вызвать какой-то метод определенного объекта. А подсказку использовать для собственного удобства (чтобы не писать руками) или для напоминания (если вы точно не помните, как называется метод).

Пример 65. Изменять имена с помощью подсказки

Допустим, вы запомнили значение одного реквизита формы в какой-то переменной, и теперь вам нужно запомнить значение другого реквизита в другой переменной.

Чтобы не повторять все заново, гораздо удобней и проще скопировать целиком всю строку и вставить ее в процедуру через буфер обмена.

В новой строке измените имя переменной, затем выделите двойным щелчком мыши имя реквизита и нажмите **Ctrl + <Пробел>**. После этого вам будет достаточно ввести первую букву (или несколько первых букв) имени нового реквизита и выбрать его из списка контекстной подсказки. После выбора новое значение целиком заменит старое (рис. 116).

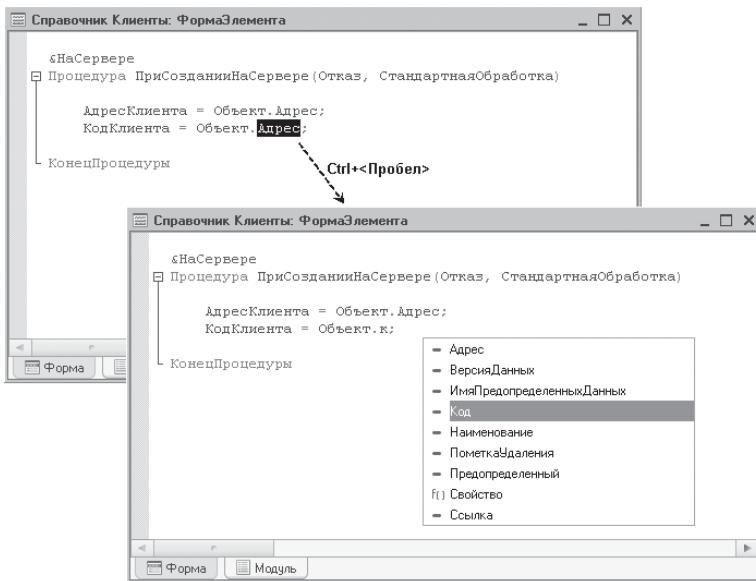


Рис. 116. Использование контекстной подсказки
при изменении имени реквизита объекта

Аналогичным образом удобно изменять значения системного перечисления. Для этого достаточно выделить старое значение, написанное после точки, нажать **Ctrl + <Пробел>** и выбрать новое значение (рис. 117).

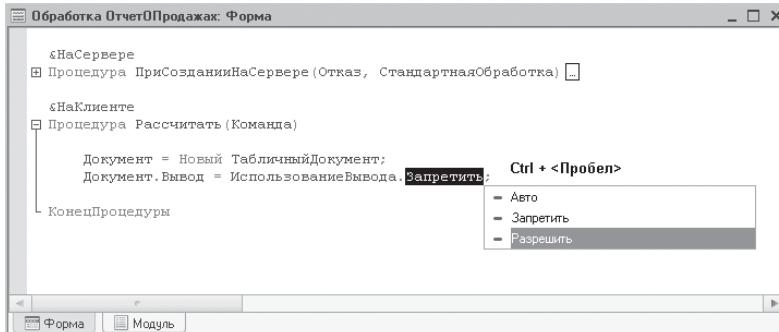


Рис. 117. Изменение значений перечисления с помощью контекстной подсказки

Пример 66. Подсказка при написании параметров

При наборе открывающих скобок и запятых открывается контекстная подсказка параметров, содержащая информацию о типах и именах передаваемых в процедуру.

Предположим, при открытии формы вам нужно показать какое-то предупреждение пользователю с помощью метода глобального контекста ПоказатьОповещениеПользователя().

Для этого вам нужно знать или вспомнить, какие параметры и какого типа надо передавать в этот метод. Но открывать для этого синтакс-помощник и искать в нем описание метода – долго и неудобно.

Наиболее эффективно воспользоваться для этого контекстной подсказкой при наборе запятой и открывающей скобки. Если соответствующая автоматическая подсказка включена, то по мере написания системной функции или метода платформа будет подсказывать вам состав и тип передаваемых параметров.

В процедуре модуля формы с помощью контекстной подсказки напишите имя метода (наберите «показать», нажмите **Ctrl + <Пробел>**). Имя метода ПоказатьОповещениеПользователя и открывающая скобка будут вставлены в текст процедуры.

После этого будет открыто окно с подсказкой типа первого параметра процедуры. Заполните этот параметр и поставьте запятую. Будет открыто окно с подсказкой второго параметра. Если этот параметр вам не важен, можете его пропустить, поставить еще раз запятую, и будет открыто окно с подсказкой следующего параметра и т. д. (рис. 118).

Имя текущего параметра выделяется жирным шрифтом. Под описанием параметра отображается тип значения (или несколько типов, перечисленных через запятую), которое можно передавать в этот параметр (см. рис. 118).

Нажав на имя типа, вы можете перейти в синтакс-помощник и прочитать информацию об этом типе. Если глав с описанием типа в синтакс-помощнике много, то будет открыто окно со списком глав. В нем вы можете найти нужную главу визуально или с помощью окна поиска (**Ctrl + F**) и открыть нужное свойство в синтакс-помощнике.

Если у метода есть несколько вариантов синтаксиса, то будет отображен первый по порядку вариант. С помощью стрелок вверх и вниз, которые будут находиться перед описанием синтаксиса, вы можете переключаться между различными вариантами.



Рис. 118. Использование контекстной подсказки при написании параметров процедуры

Если же вдруг вы, например, установили курсор в другое место модуля и окно контекстной подсказки параметров закрылось, то, чтобы вновь его открыть, вам нужно вернуться в прежнюю позицию и снова нажать **Ctrl + Shift + <Пробел>**.

Пример 67. Подсказка после кавычки

В некоторых операторах вам требуется набирать строковые константы, которые имеют формализованный вид и являются, например, полным именем объекта конфигурации или указывают на некоторую стандартную форму объекта. Например, в методе **ПредопределенноеЗначение()** вам может потребоваться набрать строку «**Справочник.СправочникСклады.ОсновнойСклад**» или в методе **ОткрытьФорму()** – строку «**Справочник.Клиенты.ФормаСписка**». При наборе таких строк вручную легко ошибиться, поэтому эффективнее пользоваться контекстной подсказкой.

Предположим, при выполнении некоторой команды вам нужно открыть форму списка справочника Клиенты с помощью функции глобального контекста **ОткрытьФорму()**.

В обработчике команды с помощью контекстной подсказки напишите имя функции (наберите «**открытьФ**», нажмите **Ctrl + <Пробел>** и выберите имя

функции ОткрытьФорму). Открывающая скобка будет поставлена автоматически.

Затем откроется окно с контекстной подсказкой параметров. В качестве параметра функции вам нужно указать имя формы, представленное в виде строки. Наберите открывающую кавычку.

Если автоматическая контекстная подсказка при наборе кавычки включена, то откроется список контекстной подсказки с именами форм для открытия. Выберите из него строку Справочник («спр» + Ctrl + <Пробел>). Нажмите Enter – точка подставится автоматически.

После этого появится список подсказки с именами справочников, определенных в конфигурации. Выберите справочник Клиенты – точка подставится автоматически.

Затем контекстная подсказка предложит вам для выбора список форм справочника. Выберите форму с именем ФормаСписка. После этого закрывающая кавычка будет поставлена автоматически (рис. 119).

Вам остается только поставить закрывающую скобку и точку с запятой.

Таким образом, в этом примере используется сразу и окно с контекстной подсказкой параметров функции, и список контекстной подсказки. Вручную делать почти ничего не пришлось. Это быстрее и удобнее, чем набирать «руками», а главное – можно быть уверенным, что ошибки в набранном тексте нет.

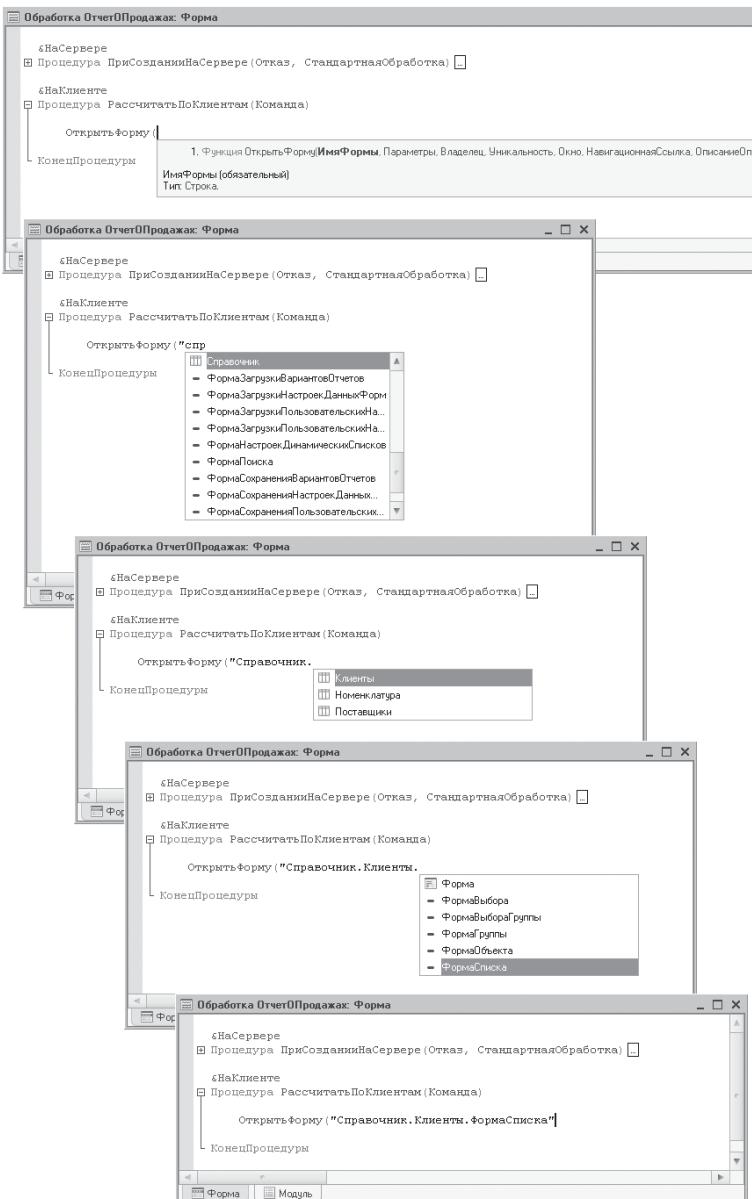


Рис. 119. Использование контекстной подсказки при написании имени открываемой формы

Пример 68. Подсказка после оператора «Новый»

Вы можете использовать контекстную подсказку при вводе оператора Новый.

Допустим, в форме обработки существует серверная процедура, в которой табличный документ заполняется данными при помощи обхода результатов выполнения некоторого запроса. Для создания объекта Запрос вам нужно использовать конструктор объекта, и результат поместить в переменную Запрос.

Для решения поставленной задачи в тексте процедуры напишите сначала имя переменной, например, Запрос. Затем наберите равно («==»).

После этого контекстной подсказки не будет, так как система «понимает», что слева от знака равенства находится имя локальной переменной пока еще неопределенного типа. Затем вручную или с помощью контекстной подсказки напишите оператор Новый.

Теперь вам нужно написать имя объекта, который вы собираетесь создавать. Чтобы не держать его в голове и не писать его «руками», лучше всего воспользоваться контекстной подсказкой. Нажмите **Ctrl + <Пробел>**.

В списке контекстной подсказки вы увидите перечисление всех программных объектов, которые вы можете создать с помощью оператора Новый. Выберите Запрос (рис. 120).

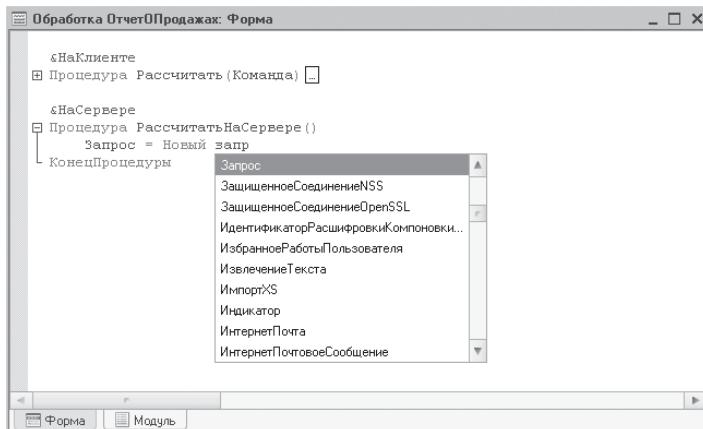


Рис. 120. Создание объекта «Запрос»

В результате выполнения этого кода созданный вами объект Запрос будет помещен в локальную переменную процедуры формы Запрос. Соответ-

ственno, эта переменная будет типизирована как объект Запрос, и ей станут доступны все свойства и методы этого объекта. После этого вы можете создавать текст запроса.

Пример 69. Настроить шаблоны текста

При редактировании модулей вам часто приходится вставлять в текст модулей различные синтаксические конструкции, функции глобального контекста, описания процедур и функций и т.п. Эти конструкции можно не писать вручную, а подставлять сразу готовые шаблоны текста, содержащие целые операторы.

Вы можете настроить возможность автоподстановки шаблонов в текст модуля в окне настройки параметров конфигуратора: Сервис > Параметры > Модули > Редактирование. Для этого вам нужно параметр Автозамена установить в значение Включить или Включить с подсказкой (рис. 121).

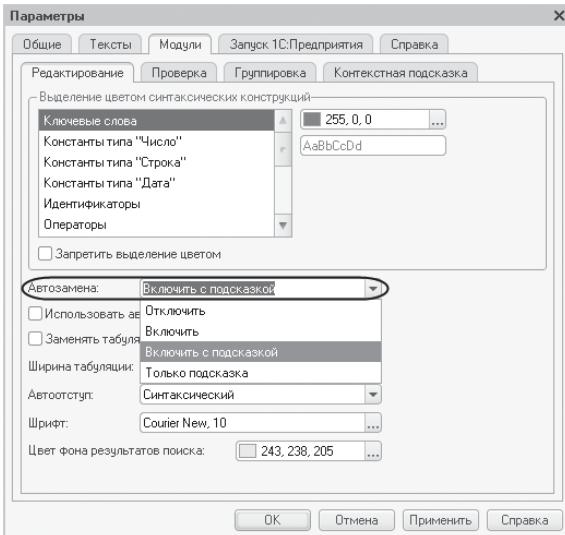


Рис. 121. Настройка параметров конфигуратора

После этого при редактировании текста модуля вам будет достаточно набрать символы, установленные для автозамены шаблона в реквизите Автоматически заменять строку (как правило, это первые четыре символа наименования шаблона), нажать клавишу Enter или Пробел, и соответствующий шаблон будет вставлен в текст модуля.

На наш взгляд, автозамена с подсказкой – это самый удобный способ, потому что он позволяет увидеть текст шаблона перед его вставкой в модуль.

Пример 70. Автоматически подставлять шаблоны

Предположим, в модуле формы документа вам нужно создать процедуру для пересчета данных по строке табличной части РассчитатьСуммуТЧ(). В эту процедуру в качестве параметра вы будете передавать переменную СтрокаТЧ, содержащую данные текущей строки табличной части, в которой необходимо выполнить пересчет.

Чтобы не набирать конструкцию <Процедура... КонецПроцедуры> вручную, легче всего воспользоваться автозаменой.

Для этого наберите «проц» и нажмите Enter. Если автозамена в модулях включена, то платформа предложит вам выбрать шаблон процедуры, которую вы хотите создать.

Выберите шаблон Процедура модуля формы. Затем выберите директиву компиляции, так как в модуле формы в явном виде указывается, в каком контексте будет выполняться та или иная процедура или функция. В данном случае пересчет данных будет выполняться на клиенте.

В следующем окне задайте имя процедуры РассчитатьСуммуТЧ. Нажмите OK (рис. 122).

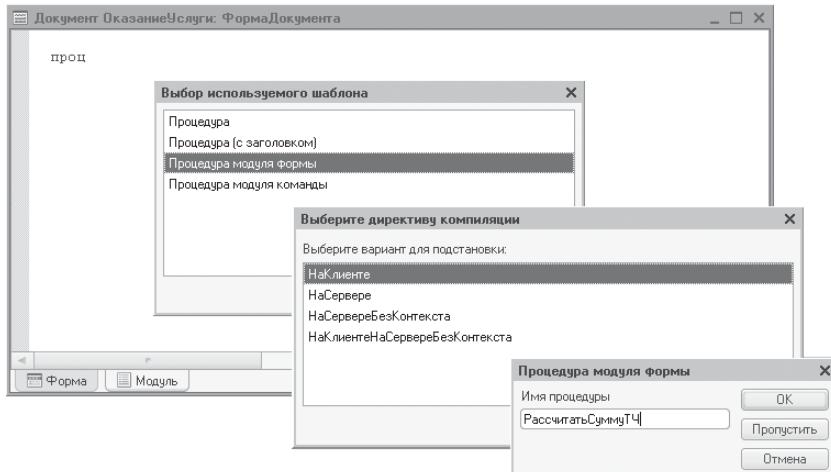


Рис. 122. Создание шаблона процедуры в модуле формы с помощью автозамены

После этого в модуле формы появится следующий текст (рис. 123).

```
// <Описание процедуры>
// Параметры:
//   <Параметр1> - <Тип.Вид> - <описание параметра>
//   <продолжение описания параметра>
//   <Параметр2> - <Тип.Вид> - <описание параметра>
//   <продолжение описания параметра>
//
// <НаКлиенте>
Процедура РассчитатьСуммуТЧ()
{
    КонецПроцедуры // РассчитатьСуммуТЧ()
}
```

Рис. 123. Текст шаблона процедуры в модуле формы

Таким образом, вы вставили в модуль формы шаблон процедуры вместе с шаблоном ее описания и директивой компиляции.

Если бы вы хотели расположить процедуру в модуле объекта или в общем модуле, то вам нужно было выбрать строку шаблонов Процедура (с заголовком). Разница лишь в том, что в этом случае вам не нужно выбирать директиву компиляции, так как в этих модулях она не требуется.

Шаблон Процедура служит для вставки в текст модуля просто объявления процедуры без описания. Этот вариант стоит использовать только в случае, когда процедура настолько простая, что не требует описания, и вы уверены, что не только вы, но и другие разработчики без труда в ней разберутся. Или же если вы разрабатываете учебную конфигурацию исключительно для себя, процедура не имеет параметров и не нуждается в контекстной подсказке.

Все вышесказанное относится также и к функциям.

Вообще же стандарты разработки в «1С:Предприятии» предполагают наличие описания процедур или функций по определенным правилам. Но стандарты – это лишь рекомендация.

Решающим моментом здесь является то, комментарии к процедуре и ее параметрам используются в контекстной подсказке. Точно так же, как для функций и процедур встроенного языка, для «самописных» процедур будет появляться контекстная подсказка параметров с описанием их типа и назначения (см. рис. 124).

Поэтому если вы хотите, чтобы контекстная подсказка работала и для вашей процедуры, нужно правильно заполнить ее описание. Если не хотите – описание можно не заполнять.

Рассмотрим пример правильного описания процедуры. Вместо конструкции <Описание процедуры> (см. рис. 123) укажите текст, описывающий назначение процедуры.

В секции Параметры: вместо конструкции <Параметр1> укажите имя параметра – СтрокаТЧ.

Вместо конструкции <Тип.Вид> укажите тип параметра – ДанныеФормыЭлементКоллекции.

Вместо конструкции <описание параметра> укажите текст, описывающий параметр.

Других параметров у этой процедуры нет.

Таким образом, текст процедуры вместе с ее описанием будет выглядеть следующим образом (листинг 4).

Листинг 4. Функция «РассчитатьСуммуТЧ()»

```
// Пересчитывает сумму по переданной строке табличной части
// Параметры:
// СтрокаТЧ – ДанныеФормыЭлементКоллекции – Данные текущей строки ТЧ
// &НаКлиенте
Процедура РассчитатьСуммуТЧ(СтрокаТЧ)

СтрокаТЧ.Сумма = СтрокаТЧ.Количество * СтрокаТЧ.Цена;

КонецПроцедуры // РассчитатьСуммуТЧ()
```

Теперь после набора имени этой процедуры и открывающей скобки будет появляться контекстная подсказка параметров с описанием их типа и назначения (рис. 124).

Из окна контекстной подсказки параметров так же, как для функций и процедур встроенного языка, вы можете нажать на имя типа и перейти в синтакс-помощник.

Таким образом, создание шаблона процедуры с описанием с помощью автозамены очень удобно и позволяет вам избавиться сразу от нескольких проблем:

- не надо вручную писать конструкцию <Процедура ... КонецПроцедуры>;
- не надо переключать регистр клавиатуры, чтобы написать директиву компиляции &НаКлиенте;
- не надо вручную или с помощью рефакторинга создавать стандартное описание процедуры, которое позволит ей появиться в контекстной подсказке.

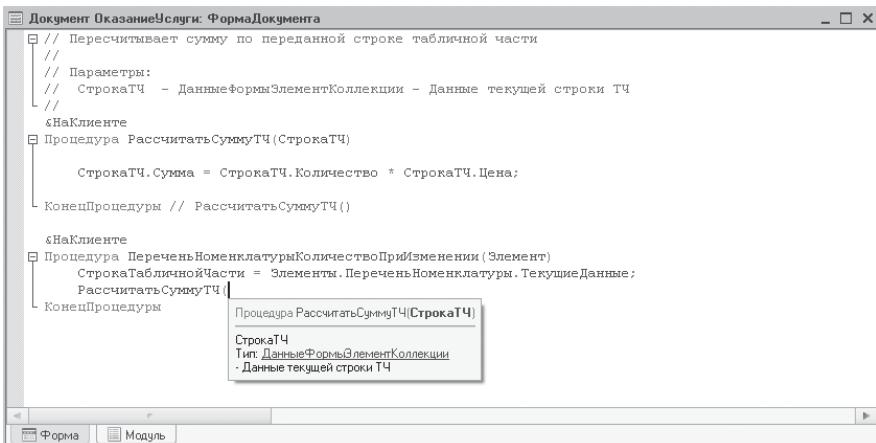


Рис. 124. Контекстная подсказка параметров созданной процедуры

ПРИМЕЧАНИЕ

Если автозамена выключена, то, чтобы вставить в модуль шаблон процедуры (или другой стандартный шаблон), вы можете также открыть окно шаблонов с помощью кнопки Открыть окно шаблонов текста (Ctrl + Shift + T) в панели инструментов конфигуратора, раскрыть группу Стандартные > Управляющие и просто перетащить его в текст модуля.

Пример 71. Создать собственные шаблоны

Помимо стандартных шаблонов, вы можете создавать и использовать свои шаблоны. Такие шаблоны особенно полезны для быстрого написания «двухязычных» конструкций, когда требуется переключаться с английского на русский регистр.

Например, вы редактируете уже существующую в конфигурации сложную процедуру. Для облегчения дальнейшей работы с ней вы хотите выделить некоторые функционально законченные области в тексте этой процедуры.

Чтобы вручную написать инструкции препроцессора, определяющие начало и конец области (#Область, #КонецОбласти), вам надо переключиться на латиницу, набрать «решетку», потом опять переключиться на кириллицу и набирать остальное.

Существующий стандартный шаблон Инструкция Область может быть не совсем удобен, так как он вставляет сразу блок <#Область ... #КонецОбласти>, в который надо потом переносить текст.

В этом случае удобнее создать и использовать собственный шаблон, который будет вставлять в текст инструкцию, определяющую только начало или только конец области.

Рассмотрим пример создания и использования собственного шаблона Начало области.

Чтобы создать файл, содержащий собственные шаблоны, в командной панели окна шаблонов (Сервис > Шаблоны текста) нажмите кнопку Действия и выполните команду Новый файл шаблонов (рис. 125).

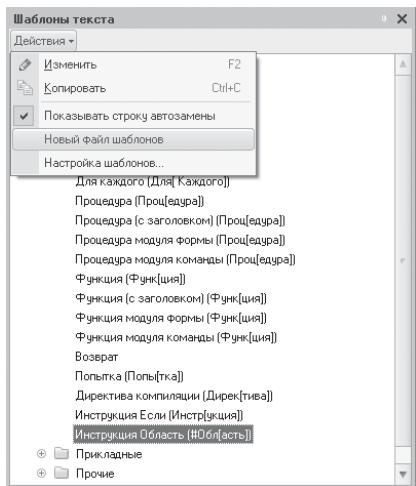


Рис. 125. Создание нового файла шаблонов

В поле Название задайте общее название своих шаблонов, например, Мои шаблоны.

Затем в командной панели окна редактирования шаблонов нажмите кнопку Добавить. В новом окне опишите шаблон инструкции препроцессору, который вы хотите создать.

В поле Название укажите название шаблона – Начало области, в поле Автоматически заменять строку задайте последовательность символов, которая будет заменяться текстом шаблона при включенном автозамене – Обла. В поле Текст шаблона укажите собственно сам текст шаблона – #Область (рис. 126).

При закрытии окна редактирования шаблона сохраните шаблон и укажите имя для нового файла шаблонов (например, МоиШаблоны.st). На вопрос конфигуратора «Добавить шаблон в список используемых шаблонов?»

ответьте утвердительно. После этого в списке шаблонов появится новая группа **Мои шаблоны**, а в ней – шаблон **Начало области** (рис. 127).

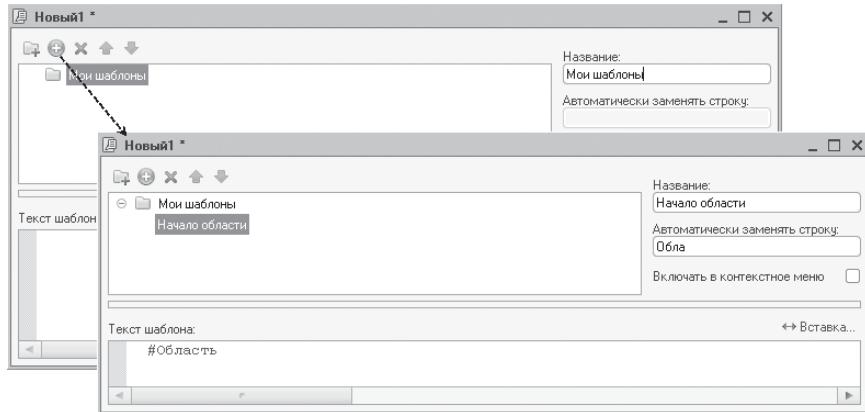


Рис. 126. Создание нового шаблона

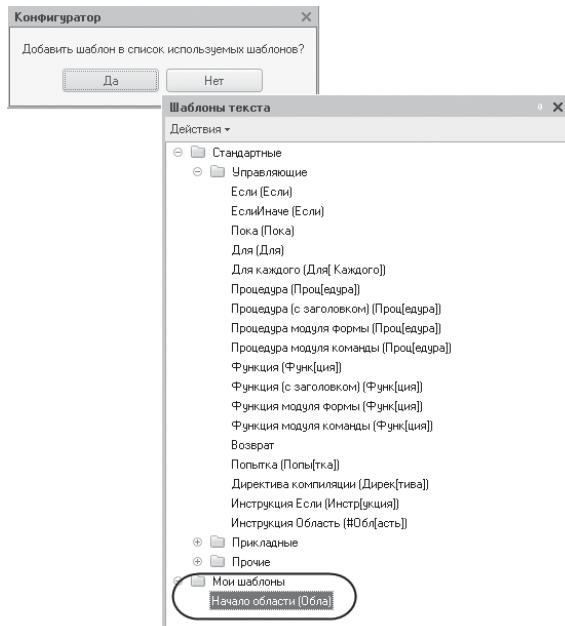


Рис. 127. Добавление своих шаблонов в список шаблонов

Если вы планируете создавать много различных шаблонов, то лучше разложить их по смысловым группам. Для этого в окне редактирования папки со своими шаблонами (**Мои шаблоны**) создайте вложенные папки командой **Добавить папку**, а в них уже добавляйте шаблоны.

Убедитесь, что все работает правильно. В модуле формы напишите «обла» – появиться всплывающая подсказка «Начало области». Нажмите клавишу Пробел. Если автозамена в модулях в параметрах конфигуратора включена, то в текст модуля подставится строка **#Область**, после которой вы можете сразу набирать имя области русскими буквами.

ПРИМЕЧАНИЕ

Кроме использования собственных шаблонов, вы можете подключить к конфигурации и другие уже созданные кем-то шаблоны. Например, для изучения книги «Практическое пособие разработчика» авторами книги были разработаны шаблоны, содержащие текст всех листингов, используемых в книге. Начинающим разработчикам мы настоятельно рекомендуем подключить и использовать их. Это упростит процесс изучения книги и позволит избежать ошибок, особенно в больших листингах.

Пример 72. Создать форматную строку

Часто бывает нужно представить пользователю какие-то значения в определенном формате. Во встроенным языке для этого используются функции: **Формат()**, **ЧислоПрописью()**, **ПредставлениеПериода()** и т. п.

Предположим, при открытии формы обработки вам нужно вывести пользователю сообщение о текущей дате. При этом представить дату вместе со временем, месяц вывести прописью, год – в четырехзначном формате и т. п. Программно это делается с помощью форматной строки, заданной в функции **Формат()**.

Запомнить, как пишутся и задаются параметры форматной строки, довольно сложно. Да это и не нужно, когда есть конструктор форматной строки. Это еще один инструмент конфигуратора, который сделает всю работу за вас.

Итак, в тексте сообщения, который вы хотите показать пользователю, наберите:

Сообщение.Текст = "Сегодня " + Формат(ТекущаяДата(),

Во второй параметр функции **Формат()** вы должны передать форматную строку, определяющую, как будет отображена текущая дата. Для этого в том месте, куда нам нужно поместить форматную строку (после запятой), выполните пункт контекстного меню Конструктор форматной строки и подтвердите создание новой форматной строки (рис. 128).

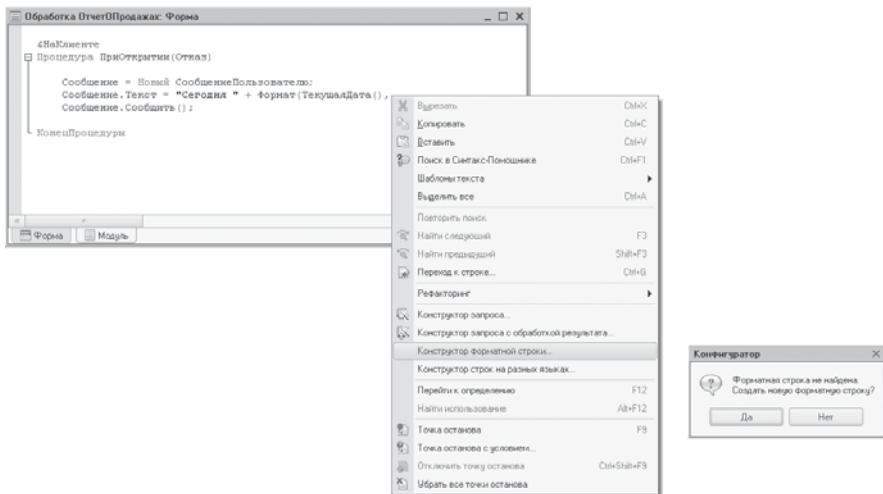


Рис. 128. Вызов конструктора форматной строки из контекстного меню

В результате откроется конструктор форматной строки. Укажите в нем параметр Язык – русский (Россия) – ru_RU, перейдите на закладку Дата и задайте параметр Локальный формат дата – DDT (отображение даты вместе со временем, месяц – прописью, год – в четырехзначном формате).

В центре окна конструктора в окне Пример показывается, как будет выглядеть в интерфейсе сформированная вами форматная строка, а внизу видите форматную строку, которая будет вставлена в текст модуля (рис. 129).

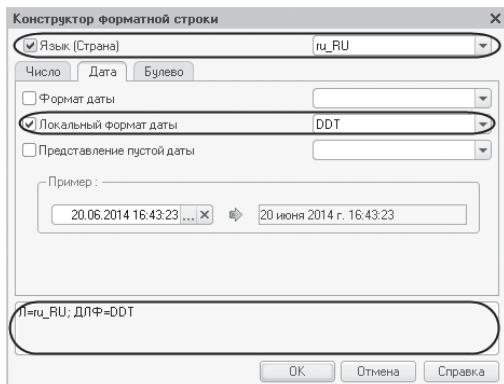


Рис. 129. Конструктор форматной строки

После нажатия ОК форматная строка будет вставлена в качестве второго параметра функции Формат(), рис. 130.

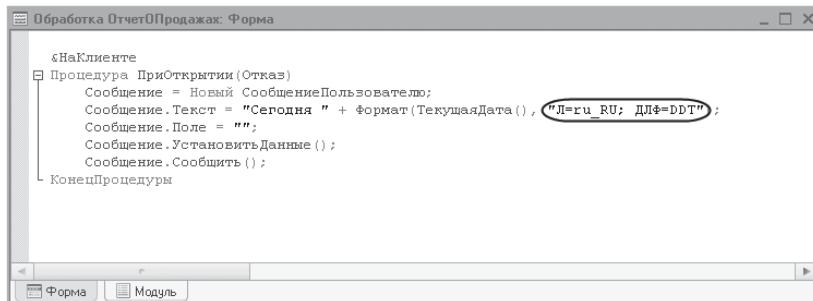


Рис. 130. Текст процедуры «ПриОткрытии» в модуле формы

В дальнейшем, установив курсор внутрь форматной строки, вы можете снова открыть конструктор форматной строки из контекстного меню, выполнив его соответствующий пункт. При этом параметры конструктора будут установлены так, как они установлены в уже существующей форматной строке (см. рис. 129). Если вы измените значения этих параметров, то существующая форматная строка будет заменена на новую.

Если же параметры форматной строки заданы неверно или форматная строка отсутствует, то конфигуратор предложит ее создать (рис. 131).

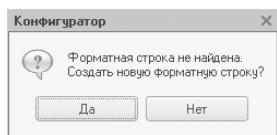


Рис. 131. Создание новой форматной строки

Таким образом, с помощью конструктора форматной строки вы можете быстро, удобно и, главное, без ошибок задать новую форматную строку или отредактировать существующую.

Пример 73. Использовать синтаксический отступ

При написании текстов модулей обычно принято использовать синтаксический отступ. Это означает, что содержимое управляющих конструкций встроенного языка выделяется символами табуляции, например, так, как это показано в приведенном ниже фрагменте модуля (листинг 5).

Листинг 5. Фрагмент процедуры

```
Если НЕ ОтветПередДобавлением Тогда
    СтандартнаяОбработка = Ложь;
    Оповещение = Новый ОписаниеОповещения("ДобавлениеЗавершение", ЭтотОбъект,
                                                ВыбранноеЗначение);
    ПоказатьВопрос(Оповещение, "Добавить номенклатуру в табличную часть?", РежимДиалогаВопрос.ДаНет);
Иначе
    Для Каждого ВыбранныйЭлемент Из ВыбранноеЗначение Цикл
        НоваяСтрока = Объект.Материалы.Добавить();
        НоваяСтрока.Материал = ВыбранныйЭлемент;
    КонецЦикла;
КонецЕсли;
```

В данном фрагменте строки модуля, расположенные внутри управляющих конструкций Если... Тогда... Иначе... КонецЕсли и Для Каждого... Из... Цикл... КонецЦикла, смещены вправо, чтобы подчеркнуть их «вложенность». Текст модуля, отформатированный с использованием синтаксического отступа, удобнее в восприятии и проще в отладке.

Сам текст процедур также должен быть сдвинут относительно начала и окончания процедур и функций, которые располагаются в крайней левой позиции в тексте модуля. Кроме того, с крайней левой позиции обычно начинаются директивы компиляции, описание процедур и функций, объявления переменных модуля и инструкции препроцессора.

Когда вы пишете какую-либо управляющую конструкцию, например, Если... Тогда, текст на следующей строке, вложенный в эту конструкцию, должен быть смещен вправо на один шаг табуляции.

Чтобы не нажимать клавишу Tab вручную на каждой строке конструкции, в настройках конфигуратора (Сервис > Параметры > Модули > Редактирование) параметр Автоотступ стандартно устанавливается в значение Синтаксический. При этом вводимый текст, расположенный внутри управляющих конструкций, после перехода на новую строку автоматически смещается вправо путем добавления в начало строки необходимого количества знаков табуляции (рис. 132).

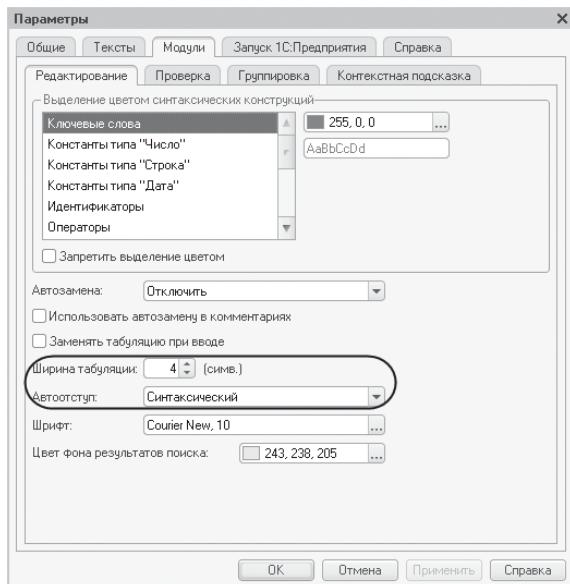


Рис. 132. Настройка параметров редактирования текстов модулей

Пример 74. Форматировать текст модуля

Предположим, вы дорабатываете незнакомую конфигурацию и в каком-то модуле вам попался неотформатированный текст. Вам трудно в нем разобраться, и вы хотите улучшить его читаемость, да и вообще сделать более «приличным».

В этом случае вам не нужно вручную двигать строки текста с помощью клавиши Tab, так как это гораздо удобнее и быстрее сделать с помощью форматирования блока текста.

Для этого выделите этот фрагмент текста (или сразу весь модуль – Ctrl + A) и нажмите кнопку Форматировать (Alt + Shift + F) в командной панели Текст. Она стандартно отображается в нижнем левом углу окна конфигуратора при редактировании любого модуля (рис. 133).

При этом система проанализирует выделенный фрагмент текста и сдвигнет вправо содержимое каждой синтаксической конструкции, используя заданный шаг табуляции. Таким образом, при формировании выделенного текста автоматически используется синтаксический отступ. Значение

параметра Автотступ на это никак не влияет – оно учитывается только при ручном вводе текста.

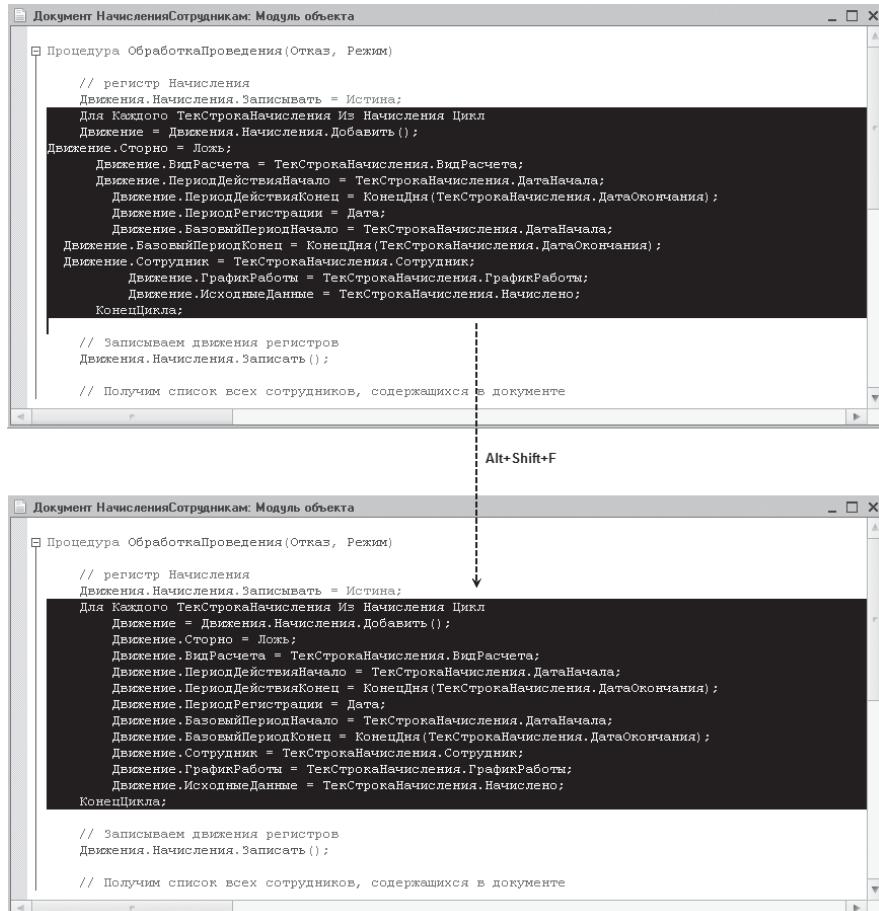


Рис. 133. Форматирование блока текста

ПРИМЕЧАНИЕ

Если отображение командной панели Текст по каким-то причинам у вас отключено, то отформатировать выделенный текст можно также с помощью команды главного меню Текст > Блок > Форматировать.

Пример 75. Закомментировать фрагмент программы

Во время разработки конфигурации, в процессе отладки модулей, часто бывает нужно закомментировать некоторые строки кода, чтобы они не исполнялись. Превратить строку кода в комментарий вы можете путем добавления в начале строки двойного слеша «//».

Вручную добавлять/удалять признак комментария (в особенности если текст большой) довольно утомительно.

Наиболее эффективно для этого выделить фрагмент текста и нажать кнопку Добавить комментарий  (Ctrl + Num + "/") в панели инструментов конфигуратора.

Чтобы снять с выделенных строк кода признак комментария и сделать их исполняемыми, вы можете нажать кнопку Удалить комментарий  (Ctrl + Shift + Num + "/") в панели инструментов.

Пример 76. Переносить длинные строки

Обычно принято не использовать в модулях длинные строки, так как их трудно читать и воспринимать. Рекомендуется строки более 120 символов переносить с помощью символа переноса «|», который пишется в начале каждой перенесенной строки. Наиболее частыми примерами использования таких длинных строк являются тексты запросов.

Например, бывает ситуация, когда запрос отложен в консоли запросов, и его надо скопировать оттуда в текст модуля. Но запрос в консоли пишется «как есть», то есть это просто текст запроса без кавычек и символов переноса строк. В программном же модуле текст запроса задается в кавычках, в виде длинной строки, используя символы «|». Например (листинг 6):

Листинг 6. Текст запроса в модуле

```
Запрос = Новый Запрос(  
    "ВЫБРАТЬ РАЗЛИЧНЫЕ  
        | НачисленияСотрудникамНачисления.Сотрудник  
    ИЗ  
        | Документ.НачисленияСотрудникам.Начисления КАК  
            | НачисленияСотрудникамНачисления  
    ГДЕ  
        | НачисленияСотрудникамНачисления.Ссылка = &ТекущийДокумент");
```

В этом случае мы рекомендуем поступать так. Скопируйте текст из консоли запросов в буфер обмена и затем поместите этот текст в модуль, в заготовку текста запроса между кавычек (Запрос = Новый Запрос("")).

После этого выделите нужные строки текста запроса и выполните команду главного меню Текст > Блок > Добавить перенос строки.

Пример 77. Создать текст запроса

Предположим, чтобы оптимизировать процедуру проведения документа, вам нужно создать в ней запрос, помещающий данные табличной части документа во временную таблицу. Данные этой таблицы будут затем использоваться другими запросами через менеджер временных таблиц.

Для этого в процедуру проведения документа поместите заготовку для создания запроса (листинг 7) и создайте текст запроса с помощью конструктора запроса.

Листинг 7. Заготовка для создания запроса

```
// Создать менеджер временных таблиц.  
МенеджерВТ = Новый МенеджерВременныхТаблиц;  
  
Запрос = Новый Запрос;  
  
// Укажем, какой менеджер временных таблиц использует этот запрос.  
Запрос.МенеджерВременныхТаблиц = МенеджерВТ;  
  
Запрос.Текст = "";
```

Поместите курсор между кавычек в строке, где устанавливается текст запроса, вызовите контекстное меню и выберите пункт Конструктор запроса (рис. 134).

Поскольку текст запроса пока отсутствует, конструктор предложит создать новый запрос. Согласитесь с этим.

На первой закладке конструктора Таблицы и поля выберите в качестве источника данных запроса таблицу ОказаниеУслугиПереченьНоменклатуры (это табличная часть документа ОказаниеУслуги) и ее поля Номенклатура, Номенклатура.ВидНоменклатуры, Количество, Сумма (рис. 135).

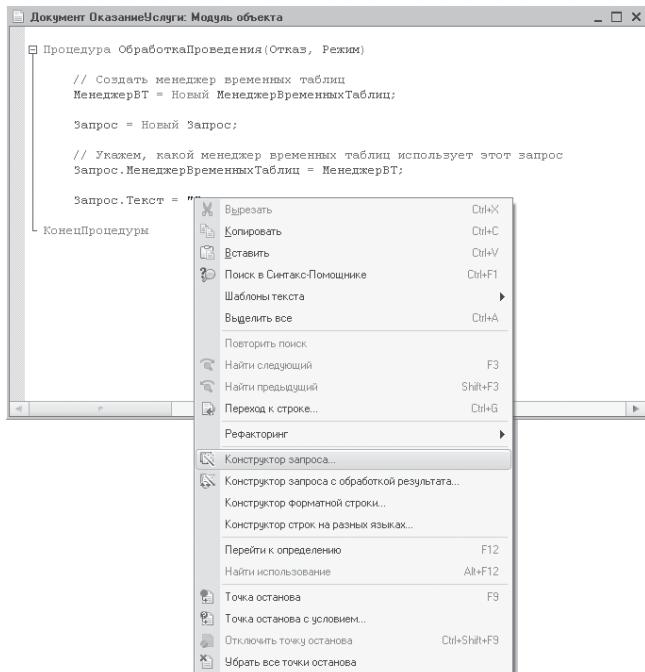


Рис. 134. Вызов конструктора запроса

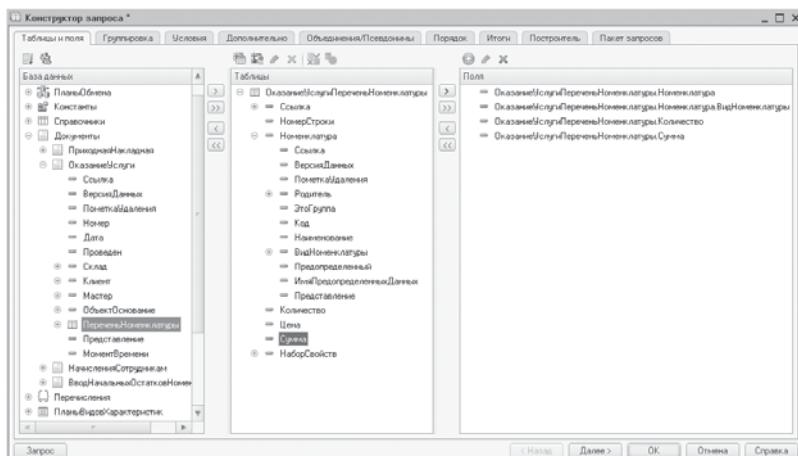


Рис. 135. Исходные данные для запроса

Затем задайте условие отбора записей из таблицы только для строк проводимого документа. Для этого перейдите на закладку конструктора Условия и перетащите в список условий поле Ссылка (рис. 136).

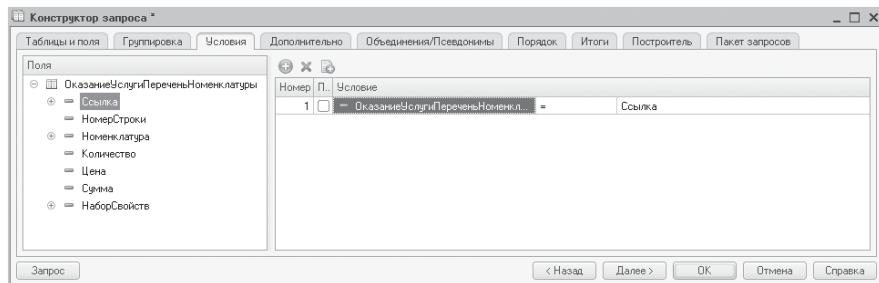


Рис. 136. Условие отбора записей из таблицы

При выполнении запроса ссылка на документ будет передана в параметр запроса Ссылка.

На закладке Группировка сгруппируйте свои записи по полю Номенклатура и Номенклатура Вид Номенклатуры. Укажите, что рассчитываться будет сумма значений для полей Количество и Сумма (рис. 137).

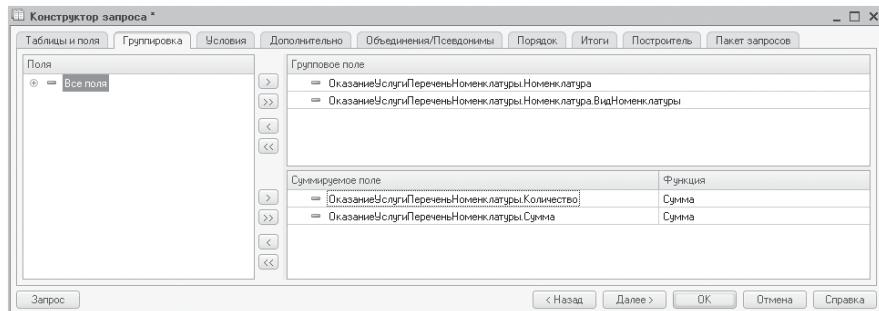


Рис. 137. Группировка записей результата запроса

В результате значения номенклатуры повторяться не будут, и для каждого из них будут посчитаны суммарные значения по полям Количество и Сумма, если в табличной части документа содержится несколько строк с одинаковой номенклатурой.

Чтобы результат запроса поместить во временную таблицу, перейдите на закладку Дополнительно и отметьте пункт Создание временной таблицы. Задайте имя временной таблицы – НоменклатураДокумента (рис. 138).

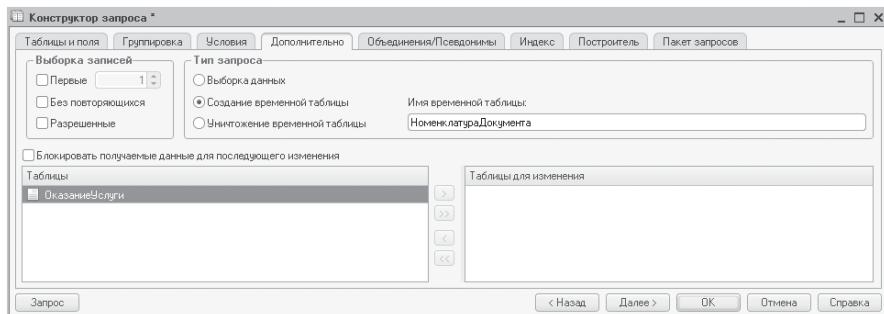


Рис. 138. Создание временной таблицы

Нажмите OK. Созданный текст запроса, разделенный символами переноса строки, будет помещен в текст модуля между кавычками (листинг 8).

Листинг 8. Текст запроса, созданный конструктором запроса

```
Запрос.Текст = "ВЫБРАТЬ
    ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
    ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры,
    СУММА(ОказаниеУслугиПереченьНоменклатуры.Количество) КАК Количество,
    СУММА(ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК Сумма
ПОМЕСТИТЬ НоменклатураДокумента
ИЗ
    Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК
        ОказаниеУслугиПереченьНоменклатуры
ГДЕ
    ОказаниеУслугиПереченьНоменклатуры.Ссылка = &Ссылка
СГРУППИРОВАТЬ ПО
    ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
    ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры";
```

Вот так легко и быстро вы создали текст своего запроса. Ко всему прочему его еще и не понадобилось форматировать (сдвигать, вставлять символы переноса строки). Все это уже «заложено» в конструкторе запроса.

Итак, мы рассмотрели простой пример, на котором показали только принцип использования конструктора запроса для создания текста запроса. Конечно, это гораздо удобнее, чем писать запрос вручную.

Пример 78. Создать запрос и обработать его результат

Конечно, чаще всего запрос создается не сам по себе, а с целью дальнейшей программной обработки данных, полученных при выполнении запроса.

Предположим, вам нужно вывести в табличный документ список всех расходных накладных, включая список товаров, проданных определенному покупателю.

Вместо того чтобы вручную писать программный код для обработки результатов запроса, гораздо проще и удобней использовать для этого Конструктор запроса с обработкой результата. Этот конструктор поможет вам не только визуально сконструировать запрос, но и создать готовый фрагмент кода для получения данных с помощью запроса и обхода выборки его результатов или вывода этих данных в табличный документ или диаграмму.

Для примера откройте серверную процедуру демонстрационной обработки, куда вы хотите поместить запрос, и вызовите из контекстного меню пункт Конструктор запроса с обработкой результата.

Подтвердите, что вы хотите создать новый запрос. После этого откроется конструктор запроса с обработкой результата. В группе Тип обработки отметьте опцию Вывод в табличный документ (рис. 139).

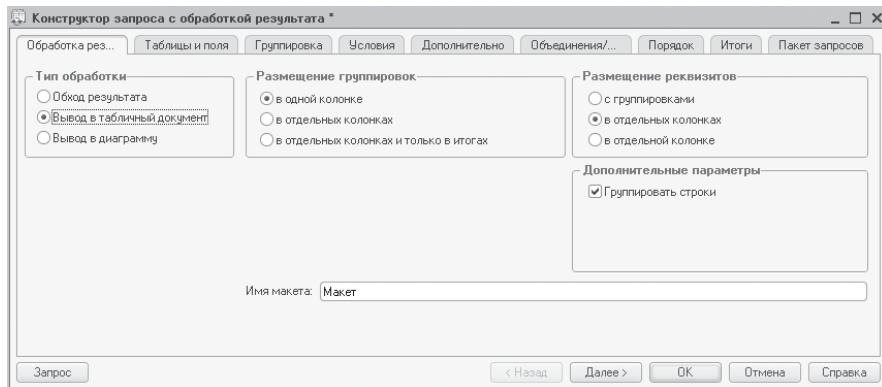


Рис. 139. Окно конструктора запроса с обработкой результата

На закладке конструктора Таблицы и поля выберите в качестве источника данных запроса таблицу РасходнаяНакладная и ее поля Дата, Номер, Склад. Затем выберите целиком табличную часть Товары и оставьте в ней только поля НомерСтрока, Номенклатура, Количество, Цена, Сумма (рис. 140).

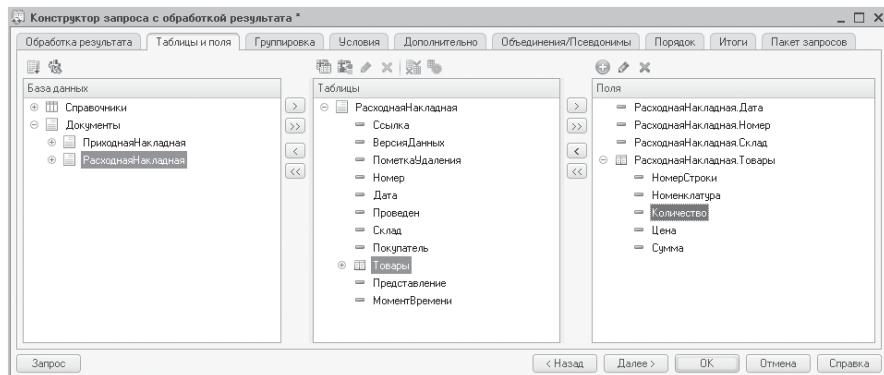


Рис. 140. Исходные данные для запроса

На закладке Условия задайте условие отбора записей из таблицы по полю Покупатель (рис. 141).

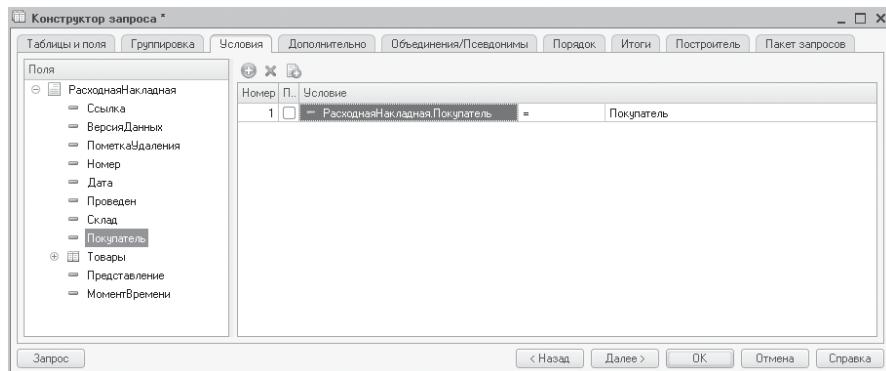


Рис. 141. Условие отбора записей из таблицы

На закладке Порядок задайте упорядочивание записей результата запроса по полю Дата (рис. 142).

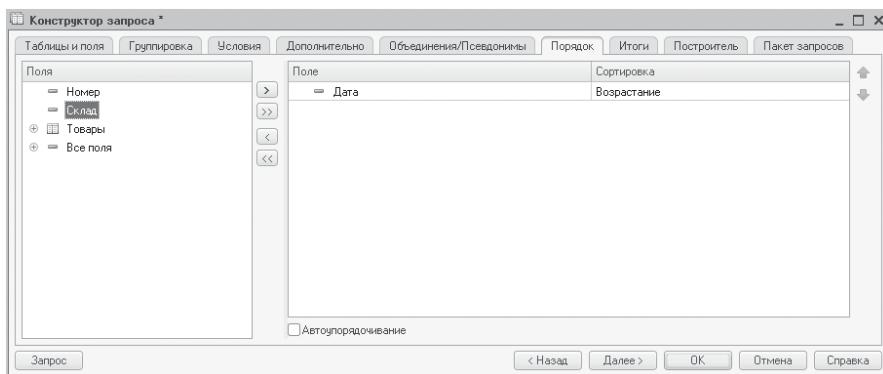


Рис. 142. Упорядочивание записей результата запроса

На этом создание запроса закончено. Нажмите OK.

Конструктор создаст макет табличного документа, в который будут выводиться данные (рис. 143).

Обработка ОтчетОПродажах: Макет						
	1	2	3	4	5	6
Заголовок	1					
	2					
	3					
ШапкаТабл:	4	Номер	Дата	Склад		
детали	5	<Номер>	<Дата>	<Склад>		
Подвалтабл	6					
ТоварыШаг	7					
	8	НомерСтроя	Номенклатура	Количество	Цена	Сумма
ТоварыДет	9	<НомерСтроя>	<Номенклатура>	<Количество>	<Цена>	<Сумма>
ТоварыПод	10					
Подвал	11					
	12					
	13					
	14					
	15					

Рис. 143. Макет табличного документа, созданный конструктором

А в тексте процедуры (откуда он был вызван) конструктор запроса с обраткой результата сформирует следующий фрагмент кода (листинг 9).

Листинг 9. Фрагмент процедуры для вывода результата запроса в табличный документ, созданный конструктором

```
// {{КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
// Данный фрагмент построен конструктором.
// При повторном использовании конструктора внесенные вручную изменения будут утеряны!!!
Макет = Обработки.ОтчетОПродажах.ПолучитьМакет("Макет");
Запрос = Новый Запрос;
```

```

Запрос.Текст =
"ВЫБРАТЬ
    РасходнаяНакладная.Номер,
    РасходнаяНакладная.Дата КАК Дата,
    РасходнаяНакладная.Склад,
    РасходнаяНакладная.Товары.(

        НомерСтрока,
        Номенклатура,
        Количество,
        Цена,
        Сумма
    )
ИЗ
    Документ.РасходнаяНакладная КАК РасходнаяНакладная
ГДЕ
    РасходнаяНакладная.Покупатель = &Покупатель

УПОРЯДОЧИТЬ ПО
    |   Дата";

```

Запрос.УстановитьПараметр("Покупатель", Покупатель);

РезультатЗапроса = Запрос.Выполнить();

```

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");
ОбластьПодвал = Макет.ПолучитьОбласть("Подвал");
ОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ШапкаТаблицы");
ОбластьПодвалТаблицы = Макет.ПолучитьОбласть("ПодвалТаблицы");
ОбластьДетальныхЗаписей = Макет.ПолучитьОбласть("Детали");
ТоварыОбластьШапкаТаблицы = Макет.ПолучитьОбласть("ТоварыШапкаТаблицы");
ТоварыОбластьПодвалТаблицы = Макет.ПолучитьОбласть("ТоварыПодвалТаблицы");
ТоварыОбластьДетальныхЗаписей = Макет.ПолучитьОбласть("ТоварыДетали");

```

```

ТабДок.Очистить();
ТабДок.Вывести(ОбластьЗаголовок);
ТабДок.Вывести(ОбластьШапкаТаблицы);
ТабДок.НачатьАвтогруппировкуСтрок();

```

ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

```

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл
    ОбластьДетальныхЗаписей.Параметры.Заполнить(ВыборкаДетальныеЗаписи);
    ТабДок.Вывести(ОбластьДетальныхЗаписей, ВыборкаДетальныеЗаписи.Уровень());

```

```

    ТабДок.НачатьГруппуСтрочек();
    ТабДок.Вывести(ТоварыОбластьШапкаТаблицы);
    ТабДок.НачатьАвтогруппировкуСтрочек();

```

ТоварыВыборкаДетальныеЗаписи = ВыборкаДетальныеЗаписи.Товары.Выбрать();

```

Пока ТоварыВыборкаДетальныеЗаписи.Следующий() Цикл
    ТоварыОбластьДетальныхЗаписей.Параметры.Заполнить(ТоварыВыборкаДетальныеЗаписи);
    ТабДок.Вывести(ТоварыОбластьДетальныхЗаписей,
                    ТоварыВыборкаДетальныеЗаписи.Уровень());
    КонецЦикла;

```

```
ТабДок.ЗакончитьАвтогруппировкуСтрочек();
ТабДок.Вывести(ТоварыОбластьПодвалТаблицы);
ТабДок.ЗакончитьГруппуСтрочек();
КонецЦикла;

ТабДок.ЗакончитьАвтогруппировкуСтрочек();
ТабДок.Вывести(ОбластьПодвалТаблицы);
ТабДок.Вывести(ОбластьПодвал);

//})КОНСТРУКТОР_ЗАПРОСА_С_ОБРАБОТКОЙ_РЕЗУЛЬТАТА
```

Как вы видите, конструктор сделал за вас практически всю работу. Осталось только передать в процедуру сам табличный документ (ТабДок), который нужно заполнить данными, и ссылку на покупателя (Покупатель) для установки параметра запроса. И все. Согласитесь, что это очень удобно!

Скорее всего, конструктор запроса с обработкой результата сэкономит время даже опытным разработчикам и поможет им избежать досадных ошибок или опечаток.

ПРИМЕЧАНИЕ

С помощью конструктора с обработкой результата вы также можете вывести результат запроса в диаграмму. А кроме этого вы можете просто создать необходимый алгоритм для программной обработки полученных результатов.

Пример 79. Проверить или изменить текст запроса

В случае если текст запроса уже существует и вам понадобилось что-то в нем изменить, часто удобнее делать это не вручную, а использовать для этого конструктор запроса.

Например, из таблицы – источника запроса выбирается много полей, которые участвуют во множестве условий и т.п. И вам нужно изменить псевдоним этой таблицы в тексте запроса.

Вручную менять псевдоним у таблицы и всех полей в тексте запроса медленно, сложно и чревато ошибками.

Гораздо эффективнее поменять его один раз в конструкторе запроса, и после этого конструктор заменит псевдонимы по всему тексту запроса, причем без ошибок.

Поместите курсор внутрь текста запроса и вызовите конструктор запроса (без обработки результата) из контекстного меню. Если текст запроса написан правильно, то конструктор «поднимет» запрос, то есть закладки конструктора заполняются исходя из уже существующего текста запроса.

Чтобы изменить псевдоним таблицы-источника, вызовите ее контекстное меню и выполните пункт **Переименовать таблицу** (рис. 144).

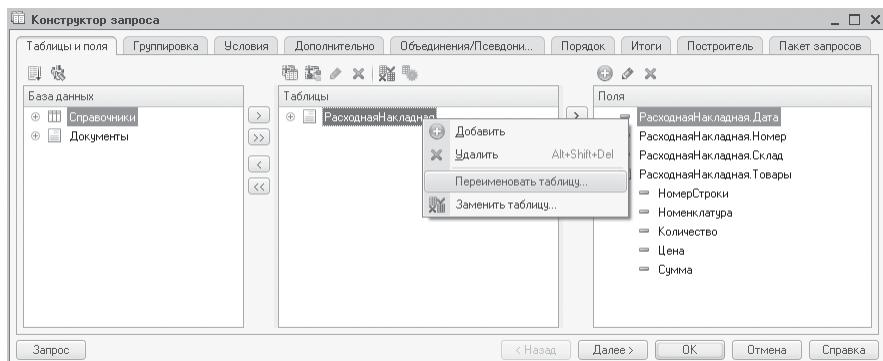


Рис. 144. Изменение псевдонима таблицы-источника в конструкторе запроса

После этого введите новое имя и нажмите **OK**.

После закрытия конструктора запроса прежний текст запроса будет заменен на новый. Псевдоним таблицы автоматически изменится по всему тексту запроса.

Приведенный пример – это не только удобный способ изменить текст запроса. Это еще и удобный способ синтаксического контроля текста запроса.

При синтаксическом контроле модулей проверяются только тексты на встроенным языке, а тексты запросов не проверяются. Потому что язык запросов это не встроенный язык, это «другой язык».

Конечно, относительно простой запрос вы можете написать и вручную. Но если вы ошиблись, то об этом станет известно только в момент исполнения запроса в режиме 1С:Предприятие.

Однако многим начинающим разработчикам хочется проверить свой запрос еще до запуска приложения. В этом как раз может помочь конструктор запроса. Если текст запроса содержит ошибки, то при установке курсора внутрь текста запроса и вызове конструктора запроса он укажет на них и не будет открыт, пока эти ошибки вручную не будут исправлены.

ПРИМЕЧАНИЕ

Если вам попался неотформатированный текст запроса (например, не разбитый на строки), то не нужно делать форматировать его вручную. Достаточно поместить курсор на этот текст и вызвать конструктор запроса. Затем просто нажать OK, и конструктор вставит текст запроса обратно в отформатированном виде.

Пример 80. Написать и отладить запрос

Иногда бывает ситуация когда, прежде чем использовать какой-то запрос в программном модуле, вам нужно убедиться, что с его помощью получаются достоверные данные. То есть вам нужно убедиться, что данные правильно отбираются по условию, группируются, суммируются и т. п.

Например, с помощью запроса вам нужно получить сложную структуру данных, но вы не знаете заранее, как это лучше сделать, и хотите попробовать разные варианты.

В этом случае удобнее всего сначала написать и отладить запрос в консоли запросов на реальных данных вашего приложения. А потом просто перенести готовый запрос в текст модуля. Консоль запросов – это внешняя обработка, которую вы можете найти в информационно-технологическом сопровождении (<http://its.1c.ru>): Разработка и администрирование > Разработка > Дополнительные средства разработки: библиотеки, обработки, руководства > Универсальные отчеты и обработки > Запросы и отчеты.

Предположим, вам необходимо узнать, в каких расходных накладных и каким покупателям продавались товары, перечисленные в составе конкретной приходной накладной. Чтобы получить эти данные, вам понадобится сложный запрос, который использует данные еще одного вложенного запроса (листинг 10).

Листинг 10. Пример запроса

```
ВЫБРАТЬ
    Расход.Ссылка КАК Документ,
    Расход.Покупатель КАК Покупатель
ИЗ
    Документ.РасходнаяНакладная КАК Расход
ГДЕ
    Расход.Состав.Товар В
    (
        ВЫБРАТЬ
            ПриходСостав.Товар КАК Товар
        ИЗ
            Документ.ПриходнаяНакладная.Состав КАК ПриходСостав
        ГДЕ
            ПриходСостав.Ссылка = &Документ
    )
```

В режиме 1С:Предприятие запустите консоль запросов, напишите в ней текст этого запроса, задайте параметр запроса и нажмите кнопку Выполнить. Результат выполнения запроса вы увидите здесь же, внизу консоли запроса (рис. 145).

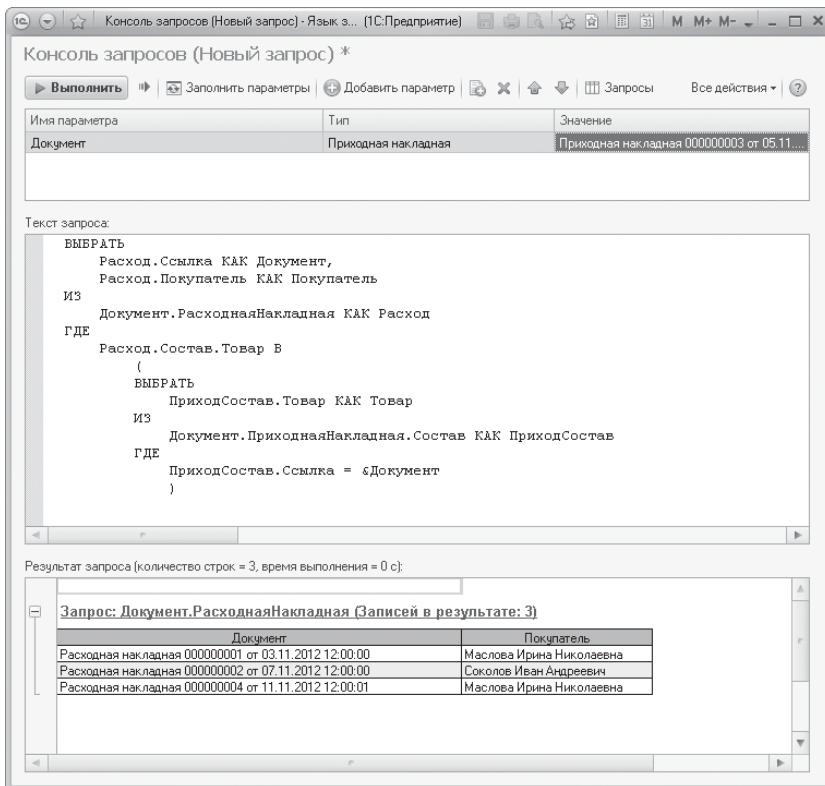


Рис. 145. Выполнение запроса в консоли запросов

Таким образом, вы можете сразу же оценить и проанализировать данные, полученные запросом.

Кстати, если вам нужно посмотреть отдельно, какие данные возвращает вложенный запрос, то для этого не понадобится удалять ненужные строки из уже написанного запроса. Достаточно выделить ту часть запроса, которую надо выполнить (в данном случае вложенный запрос), нажать кнопку Выполнить, и консоль выполнит только то, что было выделено (рис. 146).

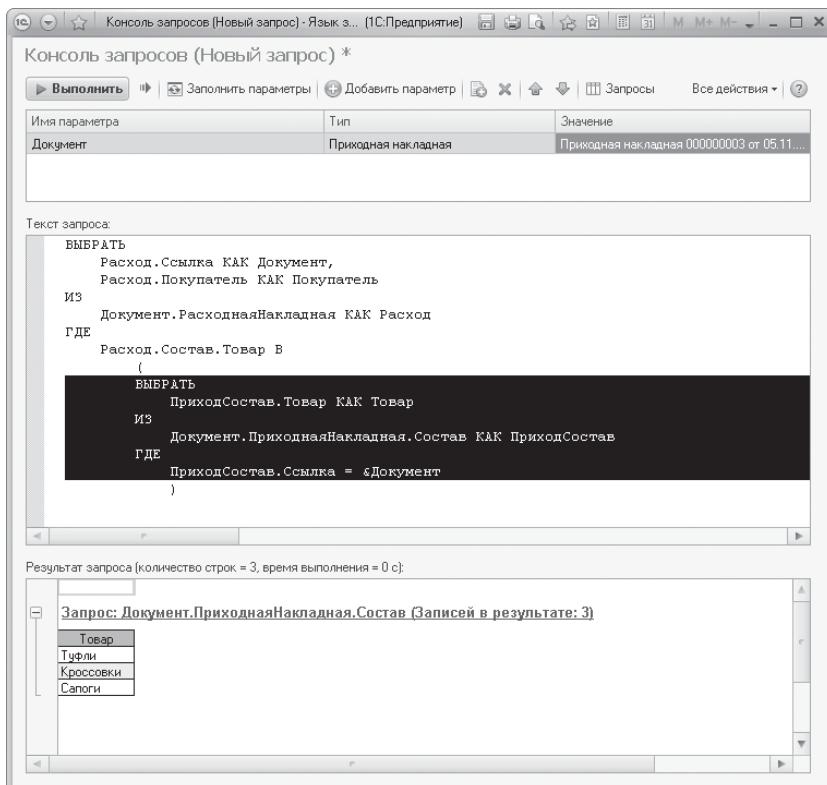


Рис. 146. Выполнение вложенного запроса в консоли запросов

Убедившись, что запрос получает правильные данные, выделите текст запроса в консоли запросов (**Ctrl + A**), поместите его в буфер обмена (**Ctrl + C**), затем перейдите в текст модуля, установите курсор между кавычек в заготовке запроса и вставьте текст из буфера обмена (**Ctrl + V**).

Осталось только в соответствии с правилами встроенного языка добавить в начало каждой строки символы переноса строки «|». Для этого выделите нужные строки текста и выполните команду **Текст > Блок > Добавить перенос строки** (листинг 11).

Листинг 11. Текст запроса в модуле

```
Запрос = Новый Запрос;
Запрос.Текст =
    "ВЫБРАТЬ
        Расход.Ссылка КАК Документ,
        Расход.Покупатель КАК Покупатель
    ИЗ
        Документ.РасходнаяНакладная КАК Расход
    ГДЕ
        Расход.Состав.Товар В
        (
            ВЫБРАТЬ
                ПриходСостав.Товар КАК Товар
            ИЗ
                Документ.ПриходнаяНакладная.Состав КАК ПриходСостав
            ГДЕ
                ПриходСостав.Ссылка = &Документ
        );
    )";
```

Пример 81. Автоматически проверять синтаксис

Автоматическая проверка синтаксиса – очень удобная возможность, которой мы рекомендуем пользоваться. Еще до исполнения текст модуля проверяется на правильность написания синтаксических конструкций встроенного языка, свойств и методов программных объектов, доступность этих объектов в определенных контекстах исполнения и т. д.

Модули управляемого приложения стандартно проверяются в двух режимах: Сервер и Тонкий клиент. Кроме того, в параметрах проверки модулей (Сервис > Параметры > Модули > Проверка) стандартно задается, что эта проверка выполняется автоматически при любом изменении и закрытии окна модуля, а также при сохранении всей конфигурации в целом.

Если при проверке модуля найдены ошибки, платформа укажет на них в специальном окне сообщений. Пока синтаксические ошибки не исправлены, код модулей не будет скомпилирован и запущен на исполнение.

Можно отключить автоматическую проверку и проверять модули вручную. Только тогда, когда вам это нужно, «по требованию». Но мы советуем использовать автоматическую проверку всегда.

Во-первых, потому, что автоматическая проверка будет запускаться при любом изменении и сохранении или закрытии модуля.

Во-вторых, получать и исправлять ошибки лучше небольшими «порциями» и «по горячим следам». Иначе потом бывает трудно вспомнить все нюансы написания кода.

ПРИМЕЧАНИЕ

Независимо от того, включена у вас автоматическая проверка синтаксиса или нет, вы всегда можете выполнить ее вручную, нажав кнопку Проверка модуля  (Ctrl + F7) в панели инструментов конфигуратора.

Пример 82. Как исправить ошибку

Этап нахождения и исправления ошибок очень важен, потому что главное – не только написать код, но и сделать его работающим. Если код не работает – значит, задача не решена. А если цель не достигнута, то и сам «процесс» вроде как ни к чему.

Начинающим разработчикам важно понимать следующее. Никто не обходится без ошибок! Поэтому когда вы столкнулись с ошибкой, главное не поддаваться эмоциям, не впадать в «ступор», не паниковать, а методично и настойчиво находить и исправлять свои ошибки.

Однако настойчивость тоже должна иметь свои пределы. Если, например, «на ночь глядя» вам кажется, что еще чуть-чуть – и все заработает, и если это «чуть-чуть» длится уже второй час, то лучше усилием воли отправить себя спать. А завтра, как говорится в одной старой книге: «Ошибка будет смотреть прямо на вас».

Ошибки бывают простые или сложные. Но последовательность действий для их исправления всегда одна и та же:

- понять, что написано в сообщении об ошибке.
- найти конкретное место в конфигурации, в котором происходит ошибка.
- посмотреть, какие значения имеют переменные в момент возникновения ошибки.

Если ошибка простая, бывает достаточно только правильно прочитать то, что написано в сообщении об ошибке. Если ошибка сложная, может понадобиться посмотреть значения переменных или даже выполнить по шагам несколько операторов, которые приводят к ошибке.

Пример 83. Внимательно анализировать текст

Значительное количество ошибок (особенно у начинающих) бывает связано с невнимательностью. Вы можете облегчить поиск таких ошибок, если будете использовать выделение идентификаторов цветом.

Предположим, вы ошиблись в строке, устанавливающей текст сообщения. И вместо имени локальной переменной Адрес написали «Адресс» (листинг 12).

Листинг 12. Процедура «АдресКлиента()

```
&НаКлиенте
Процедура АдресКлиента(Команда)
```

```
Адрес = ПолучитьАдрес(Клиент);
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = Адрес;
Сообщение.Сообщить();
```

```
КонецПроцедуры
```

При синтаксическом контроле модуля будет обнаружено имя неопределенной переменной. При этом появится окно служебных сообщений с подробной информацией о месте и причине ошибки. Из этого окна можно легко попасть к месту ошибки (рис. 147).

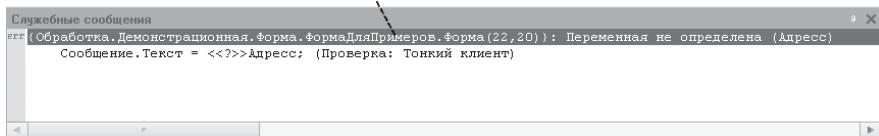
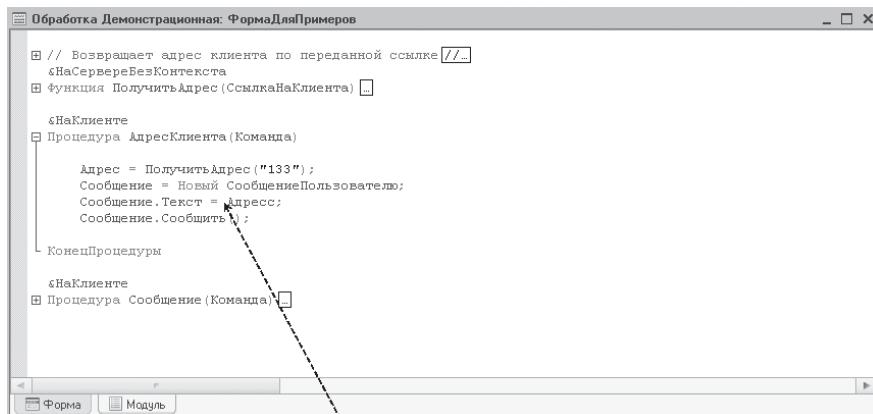


Рис. 147. Переход к строке с ошибкой из окна сообщения об ошибках

Кроме места расположения ошибки в сообщении указывается также причина ошибки – «Переменная не определена (Адрес)», из которой можно предположить, что в имени переменной допущена ошибка. Но чтобы убедиться в этом наверняка, нужно посмотреть на текст модуля.

Перейдя в строку с ошибкой, можно, конечно, визуально сравнить написанное имя с правильным именем переменной Адрес, которая определяется в процедуре двумя строчками выше. И таким образом заметить и исправить ошибку (лишнюю букву «с» в имени переменной).

Но удобнее использовать для этого выделение цветом выбранных идентификаторов, которое настраивается в параметрах конфигуратора (Сервис > Параметры > Модули > Редактирование).

В этом случае, выделив имя переменной в строке с ошибкой, вы сразу увидите, что имя переменной, определенной выше, не подсвечивается. Значит, это разные имена, и одно из них написано с ошибкой.

Бывает, что ошибка просто незаметна на глаз (например, в имени спутаны похожие символы на кириллице и на латинице – «с», «а», «т» и т.п.). В результате можно потратить много времени и нервов на поиск такой ошибки. А с помощью выделения идентификаторов цветом подобные ошибки обнаруживаются быстро.

Пример 84. Запустить отладочный сеанс от имени другого пользователя

Если в конфигурации созданы пользователи, то в конфигураторе вы обычно работаете от имени пользователя с полными правами, например, Администратор (рис. 148).

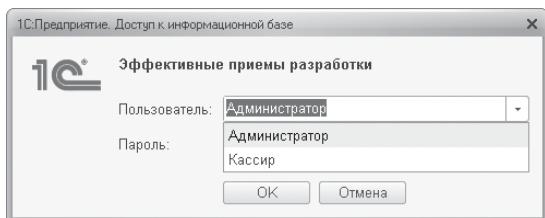


Рис. 148. Выбор пользователя при запуске конфигуратора

Чтобы проверить свои изменения, из конфигуратора вы запускаете сеанс «1С:Предприятия». Стандартно сеанс «1С:Предприятия» запускается от имени того пользователя, под которым вы вошли в конфигуратор (в нашем примере от имени администратора).

Предположим, теперь вам нужно запустить «1С:Предприятие» от имени другого пользователя, например, Кассир, чтобы проверить, как приложение работает именно под его правами.

Каждый раз отдельно запускать «1С:Предприятие», выбирать в окне запуска информационную базу, авторизоваться как пользователь Кассир – долго и неудобно. А если вы хотите что-то отладить, то вам просто необходимо запустить отладочный сеанс из конфигуратора от имени кассира.

В этом случае гораздо проще и быстрее в конфигураторе настроить нужные параметры запуска сеанса «1С:Предприятие» и запускать сеанс одним нажатием.

Для этого в настройках параметров конфигуратора (Сервис > Параметры > Запуск 1С:Предприятия > Основные) выделите пункт Имя в группе параметров Пользователь и выберите нужную строку из списка пользователей конфигурации (рис. 149).

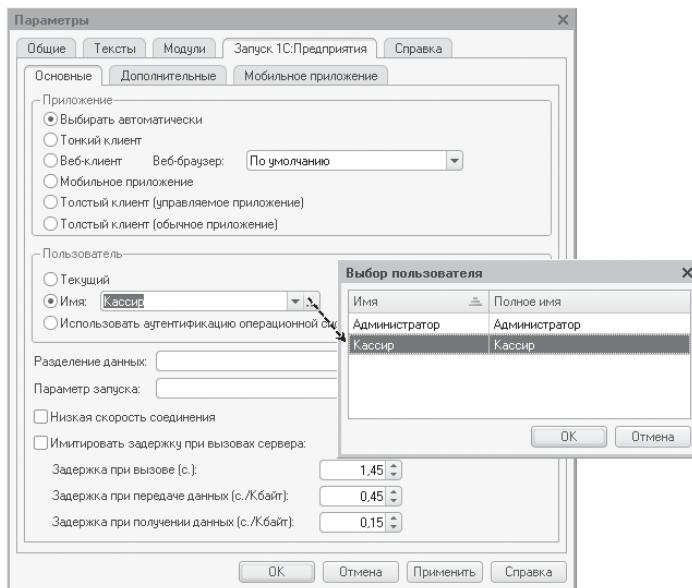


Рис. 149. Выбор пользователя, от имени которого будет запускаться «1С:Предприятие»

После этого при запуске сеанса «1С:Предприятия» из конфигуратора нажатием **Ctrl + F5** или запуска отладочного сеанса клавишей **F5** приложение будет запущено от имени выбранного в настройках параметров пользователя (рис. 150), в то время как сам конфигуратор был запущен от имени администратора.

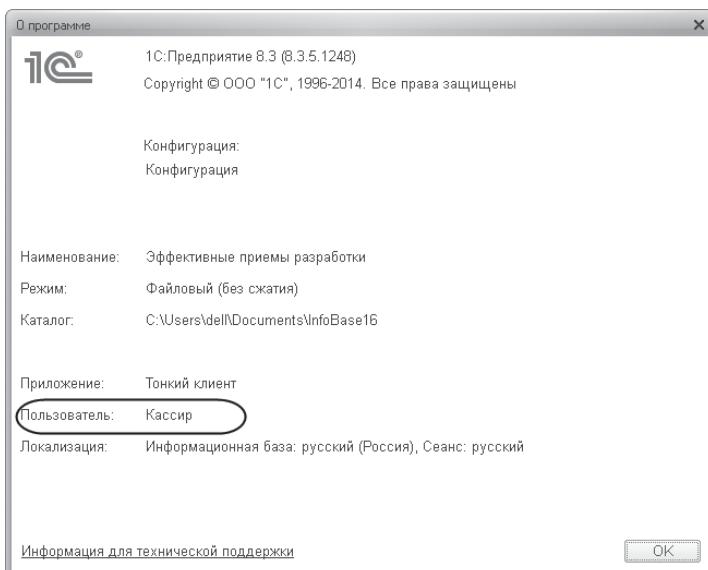


Рис. 150. Окно информации о программе в режиме «1С:Предприятие»

Пример 85. Подключить отладчик к работающему сеансу

Иногда ошибка в приложении возникает в неподходящий момент, когда вы не собирались ничего отлаживать. Просто запустили приложение для работы в режиме 1С:Предприятие и стали выполнять какую-то длинную цепочку операций. И тут неожиданно натолкнулись на ошибку. Эту ошибку хорошо бы было посмотреть в конфигураторе на тех данных, которые у вас сейчас получились. Но он не запущен.

Если сейчас закрыть приложение, запустить конфигуратор и из него – отладочный сеанс, то всю длинную последовательность действий, которую вы перед этим делали в режиме 1С:Предприятие, вам придется выполнять заново, чтобы прийти к тому состоянию приложения, в котором произошла ошибка.

В этом случае есть удобная возможность подключить конфигуратор к уже работающему сеансу, снова нажать кнопку, после которой появилась ошибка, и перейти в конфигуратор, чтобы исправить эту ошибку.

Рассмотрим самый простой случай, когда вам требуется подключиться к «1С:Предприятию», работающему на том же компьютере, что и конфигуратор, в файловом варианте работы.

Чтобы это сделать, сначала разрешите отладку в работающем клиентском приложении. Для этого из главного меню 1С:Предприятия вызовите окно настройки параметров (Главное меню > Сервис > Параметры) и установите флажок Отладка в текущем сеансе разрешена (рис. 151).

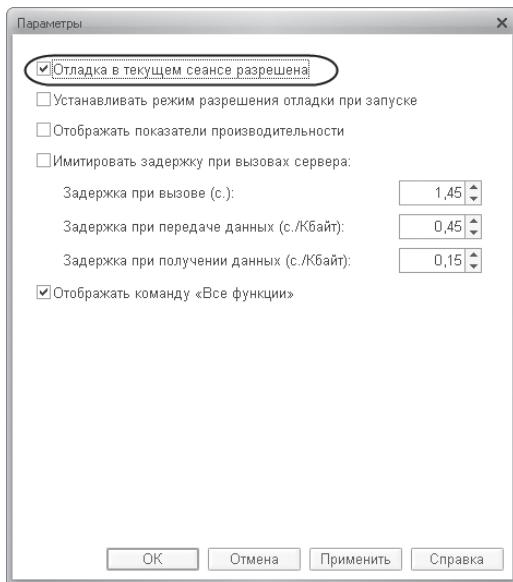


Рис. 151. Установка параметров в режиме «1С:Предприятие»

После этого запустите «1С:Предприятие» и откройте информационную базу, которую требуется отлаживать, с помощью конфигуратора.

В конфигураторе откройте окно подключения к предметам отладки, выполнив команду Отладка > Подключение. Затем поочередно выделите и подключите все доступные предметы отладки в верхнем списке с помощью кнопки Подключить. После этого предметы отладки будут перенесены из списка доступных в список подключенных предметов отладки (в нижний список), рис. 152.

В результате при появлении ошибки в «1С:Предприятии» появится окно с сообщением об ошибке. Из этого окна вы сможете уже автоматически перейти в конфигуратор и посмотреть, в чем дело.

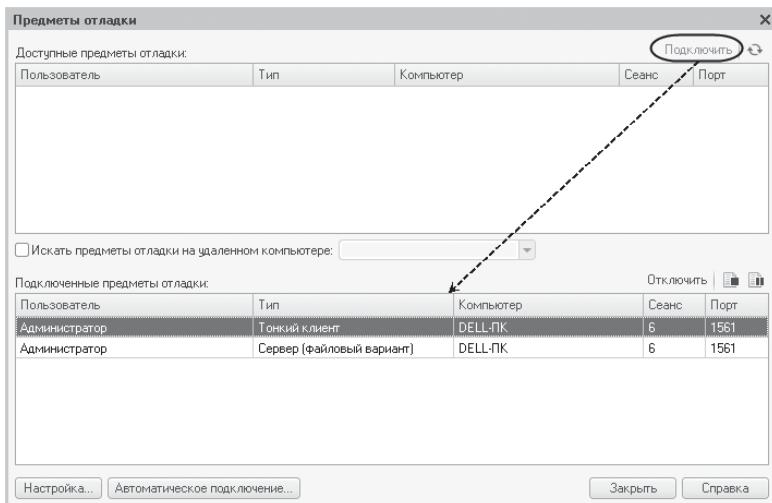


Рис. 152. Подключение предметов отладки

Пример 86. Узнать значения переменных

Пожалуй, самая распространенная ситуация, когда прикладное решение запущено в отладочном режиме из конфигуратора, чтобы протестировать и отладить его работу. Во время выполнения приложения произошла какая-то ошибка, после которой оно было остановлено, и вы вернулись в конфигуратор в то конкретное место, где возникла эта ошибка.

Далеко не всегда можно понять причину ошибки из самого сообщения. Как правило, требуется посмотреть, какие значения имеют переменные в момент возникновения ошибки.

Для этого нужно остановить выполнение приложения непосредственно перед той строкой модуля, где возникает ошибка, с помощью точки останова.

Рассмотрим конкретный пример.

Предположим, в процедуре для подбора номенклатуры в табличную часть документа открывается форма выбора справочника, в которую передается структура параметров, определяющих поведение этой формы. Предположим, вы ошиблись и во втором параметре метода ОткрытьФорму() вместо имени структуры ПараметрыФормы, которая определяется строчкой выше, написали имя Параметры (листинг 13).

Листинг 13. Обработчик команды «Подбор»

```
&НаКлиенте
Процедура Подбор(Команда)
```

```
ПараметрыФормы = Новый Структура("ЗакрыватьПриВыборе, МножественныйВыбор",
Ложь, Истина);
ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", Параметры,
Элементы.Материалы);
```

```
КонецПроцедуры
```

При открытии приходной накладной и нажатии кнопки Подбор в режиме 1С:Предприятие появится сообщение об ошибке – «Несоответствие типов (параметр номер 2)». При нажатии кнопки Подробно будет открыто окно с указанием конкретного места возникновения ошибки. При нажатии кнопки Конфигуратор вы вернетесь к месту ошибки – в модуль формы приходной накладной, к методу ОткрытьФорму().

Чтобы понять, в чем причина ошибки, вы должны посмотреть, какой тип у второго параметра (Параметры) метода ОткрытьФорму() в момент его вызова.

Для этого в модуле формы документа в строке с ошибкой (выделенной жирным шрифтом в листинге 13), на которой вы в данный момент находитесь, нужно установить точку останова.

Проще всего это сделать с помощью двойного щелчка мыши в служебной области модуля слева от нужной строки (рис. 153).

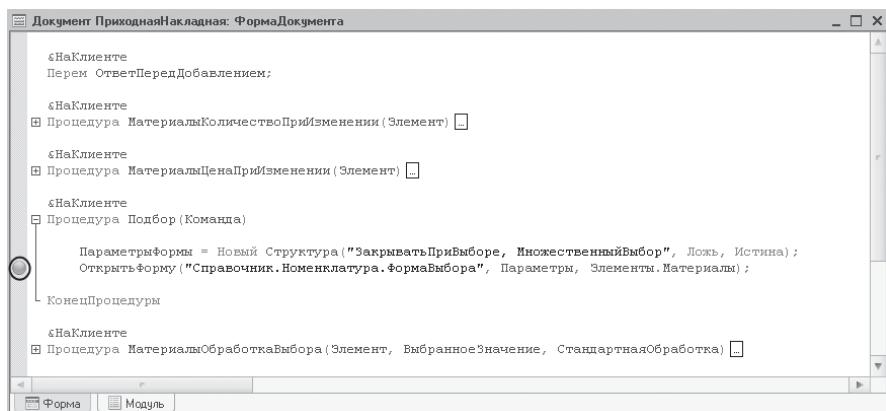


Рис. 153. Установка точки останова в тексте модуля

После этого нажмите кнопку Продолжить отладку (F5) в панели инструментов конфигуратора для продолжения отладки и подтвердите, что требуется перезапустить «1С:Предприятие».

При нажатии кнопки Подбор в форме приходной накладной исполнение приложения будет остановлено и передано в отладчик. В конфигураторе будет открыта процедура Подбор() в точке останова. Рядом с ней появится стрелка, указывающая на текущую исполняемую строку модуля (рис. 154). Заметьте, что выполнение приложения останавливается в тот момент, когда строка, содержащая точку останова, еще не выполнялась.

Теперь вам нужно узнать, какое значение и какого типа содержится в переменной Параметры в момент остановки программы. Двойным щелчком мыши выделите имя переменной и нажмите кнопку Вычислить выражение (Shift + F9) на панели инструментов конфигуратора.

Откроется окно Выражение, в котором поле Выражение заполнится именем переменной Параметры. В соответствующих колонках вы увидите значение и тип этой переменной – ДанныеФормаСтруктура. Раскрыв эту структуру, вы можете увидеть значение и тип параметров формы документа Ключ и КлючНазначенияИспользования (рис. 154).

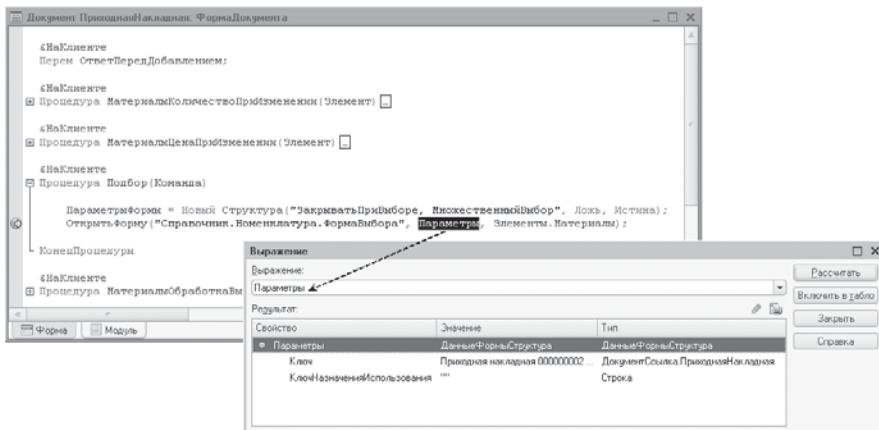


Рис. 154. Просмотр значения параметра в окне «Выражение»

Таким образом, становится понятно, что в метод ОткрытьФорму() во втором параметре передается не структура ПараметрыФормы (определенная строчкой выше), а набор параметров формы самого документа Параметры типа ДанныеФормаСтруктура. Поэтому возникает ошибка.

Бывает так, что переменная, значение которой вы хотите просмотреть, является коллекцией нескольких значений. Например, массивом или структурой. Тогда в отладчике вы можете посмотреть, из каких значений состоит этот массив или структура.

Например, вы хотите узнать, из каких значений состоит массив МассивДляРасчета. Двойным щелчком мыши выделите имя массива МассивДляРасчета в строке останова и нажмите кнопку Вычислить выражение (Shift + F9) на панели инструментов конфигуратора.

Поскольку массив – это коллекция значений, то для ее просмотра в открывшемся окне Выражение выделите имя коллекции в таблице Результат и нажмите кнопку Показать значение в отдельном окне (F2) . Так как массив содержит еще две вложенные коллекции, структуру и таблицу значений, то для их просмотра вы также можете воспользоваться клавишей F2 (рис. 155, 156).

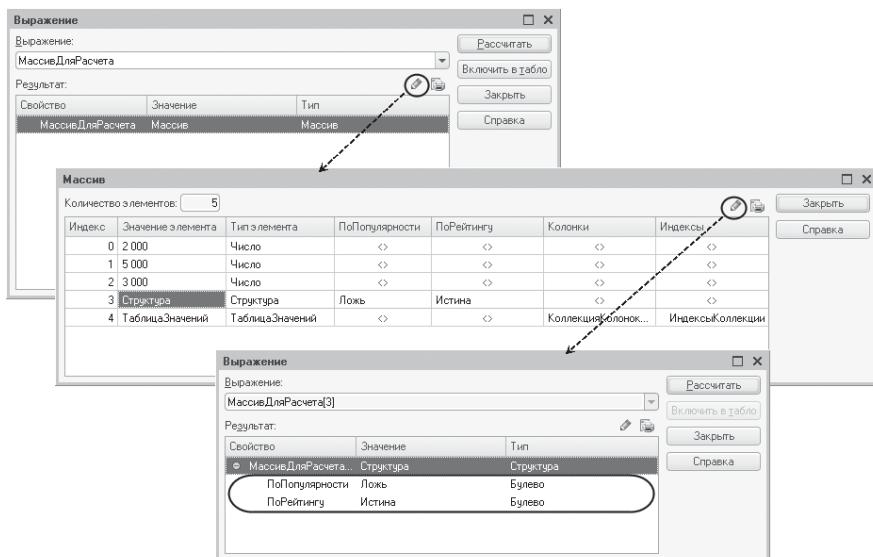


Рис. 155. Просмотр значений коллекции в окне «Выражение»

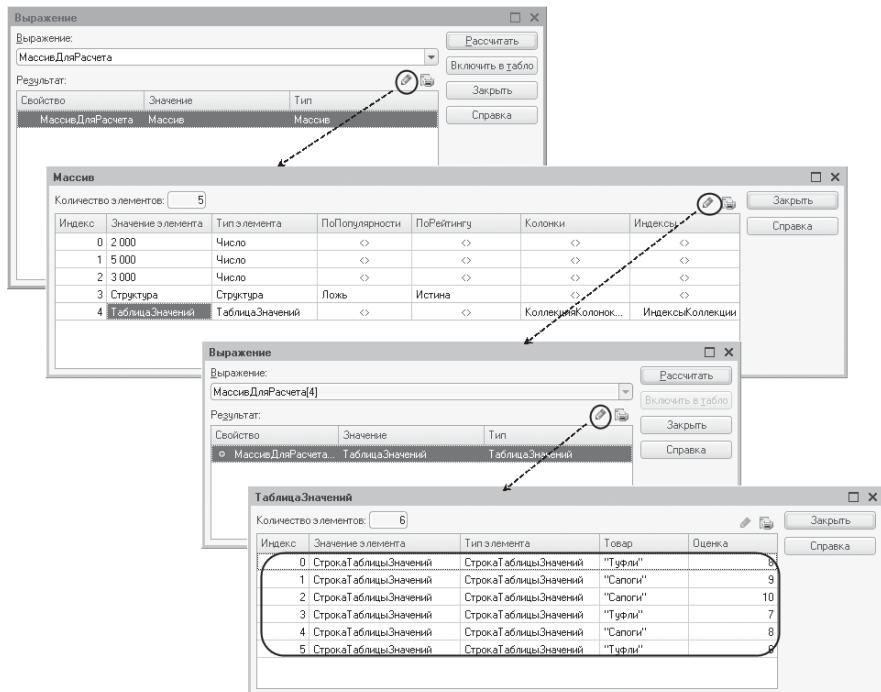


Рис. 156. Просмотр значений коллекции в окне «Выражение»

ПРИМЕЧАНИЕ

Поскольку в момент ошибки вы возвращаетесь в конфигуратор сразу на строку с ошибкой, то, чтобы поставить в этой строке точку останова, можно нажать кнопку Точка останова (F9) на панели инструментов конфигуратора. При этом не надо визуально следить, в какой строке находится курсор, чтобы щелкнуть мышью слева от него.

Пример 87. Остановить исполнение до того, как произойдет ошибка

Бывают ситуации, когда при возникновении ошибки во время исполнения «1С:Предприятия» приложение аварийно завершается. И точно определить строку, на которой происходит ошибка, невозможно. Либо существует какая-то «плавающая» ошибка, которая воспроизводится не каждый раз.

В этом случае есть возможность сделать так, чтобы приложение остановилось непосредственно перед возникновением ошибки и позволило бы вам перейти в конфигуратор.

Для этого установите флагок Останавливаться по ошибке в окне Остановка по ошибке, вызываемом командой главного меню Отладка > Остановка по ошибке (рис. 157).

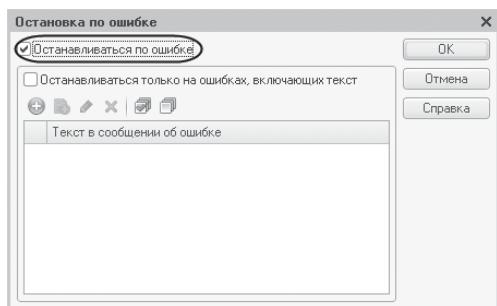


Рис. 157. Окно «Остановка по ошибке»

В этом случае выполнение приложения будет прервано непосредственно перед возникновением ошибки, откроется конфигуратор, и будет выведена информация о месте и причине ошибки (рис. 158).



Рис. 158. Ошибка времени выполнения

А после нажатия OK вы сможете посмотреть в конфигураторе, например, значения переменных, которые могли привести к такой ошибке.

Пример 88. Узнать значения нескольких переменных одновременно

Иногда бывает удобно не просто посмотреть значение какой-то одной переменной, а видеть значения сразу нескольких переменных или выражений в момент остановки приложения.

Например, вы хотите проконтролировать правильность работы функции, в которой производится разбор строки на части. Эта функция сначала находит позицию разделителя в строке, затем вырезает начало строки, потом находит следующую позицию разделителя и т. д. Проверяя правильность такого алгоритма, вам было бы удобно видеть одновременно, в какой позиции найден разделитель, какая часть отрезана, какая осталась и т. д.

Можно посмотреть по отдельности значения всех этих переменных в окне Выражение, но это долго и неудобно.

Лучше всего для этого использовать специальное окно Табло, которое позволяет в течение всего процесса отладки просматривать значения сразу нескольких интересующих вас выражений.

Например, поместите точку останова в конец функции для разбора строки с информацией о номенклатуре. В момент остановки приложения откройте Табло, нажав кнопку Табло (`Ctrl + Alt + W`)  на панели инструментов конфигуратора.

Чтобы поместить интересующее вас выражение в табло, выделите его в модуле и перетащите в табло. Поместите таким образом в табло все выражения, которые нужны вам для разбора строки, а также получившиеся в результате данные о номенклатуре (рис. 159).

В результате в момент остановки приложения сразу в одном окне вы сможете увидеть поэтапно, как разбирается строка с данными номенклатуры (рис. 160). И если надо, вы сможете легко увидеть и устраниТЬ ошибку (например, неправильно вырезалась строка для поиска).

Удобно, что окно табло остается открытым в конфигураторе до тех пор, пока вы сами его не закроете. Поэтому вы можете один раз добавить в него интересующие вас переменные, а затем только просматривать их значения каждый раз, когда вы останавливаете исполнение программы.

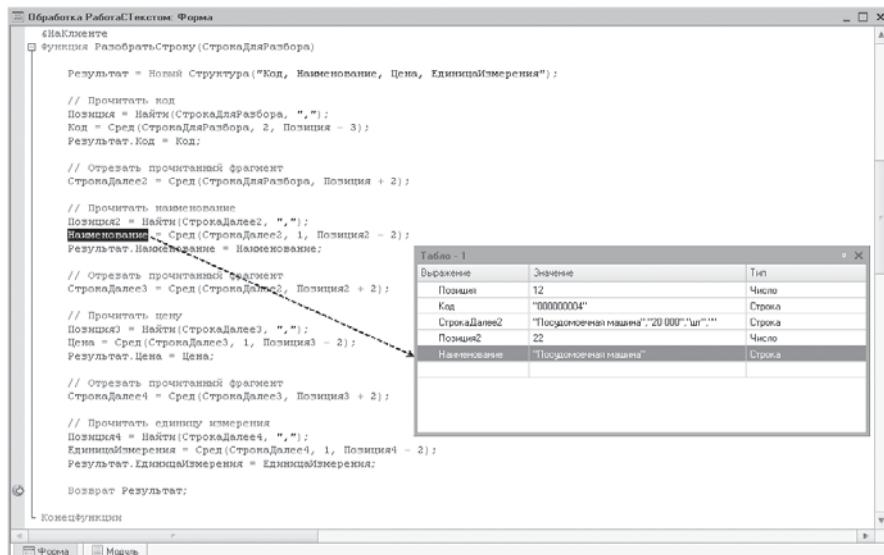


Рис. 159. Ввод выражения в табло

Выражение	Значение	Тип
Позиция	12	Число
Код	"00000002"	Строка
Строка.Далее2	"Холодильник","25 000","шт",""	Строка
Позиция2	13	Число
Наименование	"Холодильник"	Строка
Строка.Далее3	"25 000","шт,""	Строка
Позиция3	8	Число
Цена	"25 000"	Строка
Строка.Далее4	"шт,""	Строка
Единица.Измерения	"шт"	Строка

Рис. 160. Просмотр выражений в табло

Если табло для отладки вам больше не нужно, то можно очистить содержимое строк табло нажатием клавиши Del и закрыть табло.

Пример 89. Остановить исполнение при выполнении некоторого условия

Бывает так, что во время работы приложения ошибка возникает не всегда, а лишь в определенной ситуации, при обработке каких-то определенных данных.

Предположим, в цикле вы загружаете данные от разных организаций. При этом загрузка данных от большинства организаций проходит нормально, а от какой-то одной организации данные загружаются неправильно. Чтобы разобраться, в чем дело, вам нужно проанализировать момент загрузки данных от этой «подозрительной» организации.

Просто установить точку останова внутри цикла в данном случае неэффективно. Потому что вам придется останавливаться на каждом шаге цикла, смотреть значение переменной и продолжать отладку нажатием F5. Это, мягко говоря, «утомительно», ведь цикл может быть очень большим.

Гораздо эффективнее использовать для этих целей точку останова с условием. Она позволит вам прервать выполнение программы именно тогда, когда начнут обрабатываться данные «подозрительной» организации.

Для этого внутри цикла обхода элементов справочника Организации установите курсор на интересующую вас строку цикла и нажмите кнопку Точка останова с условием  на панели инструментов конфигуратора.

Затем в появившемся окне Условие останова укажите условие, при котором исполнение кода будет останавливаться. Например, вы знаете наименование этой «подозрительной» организации. Тогда можно написать: Организация.Наименование = "ООО Комфорт" (рис. 161).

В результате точка останова с условием сработает только тогда, когда будут загружаться данные организации ООО «Комфорт». На других итерациях цикла остановки не будет.

Если вы хотите изменить условие останова, установите курсор в любое место строки, содержащей точку останова с условием, и снова нажмите кнопку .

Если вы не хотите писать вручную имена переменных, объектов и т.п., содержащихся в условии останова, вы можете копировать их прямо из модуля через буфер обмена.

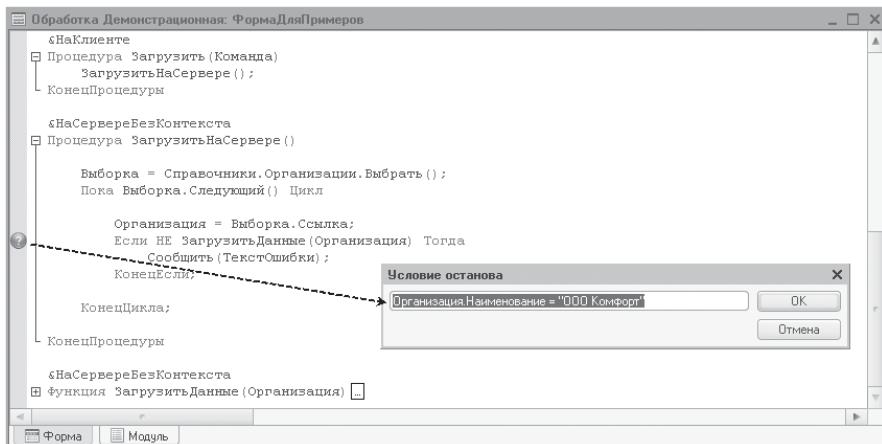


Рис. 161. Установка условия останова

Пример 90. Посмотреть изменение переменных по шагам

В процессе отладки реальной конфигурации неизбежно появляются сложные логические ошибки, найти и понять которые не так-то легко. В этом случае вам может помочь пошаговое выполнение отладки. При этом вы можете поставить точку останова за много шагов до возникновения такой ошибки и, пошагово выполняя отладку, выявить ошибку, которая не лежит на поверхности.

Например, при разборе строки с данными номенклатуры на части вы хотите пошагово проконтролировать, в какой позиции найден разделитель, какая часть строки отрезана, какая осталась и т. д.

Рассмотрим функцию для разбора строки с данными номенклатуры, в которой строка сканируется в цикле, а строки между разделителями вырезаются и добавляются в поля структуры ДанныеНоменклатуры (листинг 14).

Листинг 14. Функция для разбора строки с данными номенклатуры

```

&НаКлиенте
Функция РазобратьСтроку(СтроДляРазбора)

    ДанныеНоменклатуры = Новый Структура();

    Позиция = Найти(СтроДляРазбора, ",");
    СтроДалее = СтроДляРазбора;

```

```

Индекс = 1;
Пока Позиция <> 0 Цикл
    Если Индекс = 1 Тогда
        Значение = Сред(СтроКадалее, 2, Позиция - 3);
    Иначе
        Значение = Сред(СтроКадалее, 1, Позиция - 2);
    КонецЕсли;
    ДанныеНоменклатуры.Вставить("Поле" + Строка(Индекс), Значение);
    СтроКадалее = Сред(СтроКадалее, Позиция + 2);
    Позиция = Найти(СтроКадалее, ",");
    Индекс = Индекс + 1;
КонецЦикла;

Возврат ДанныеНоменклатуры;

КонецФункции

```

В цикле поставьте точку останова на строку СтроКадалее = ... и запустите «1С:Предприятие» в режиме отладки (рис. 158).

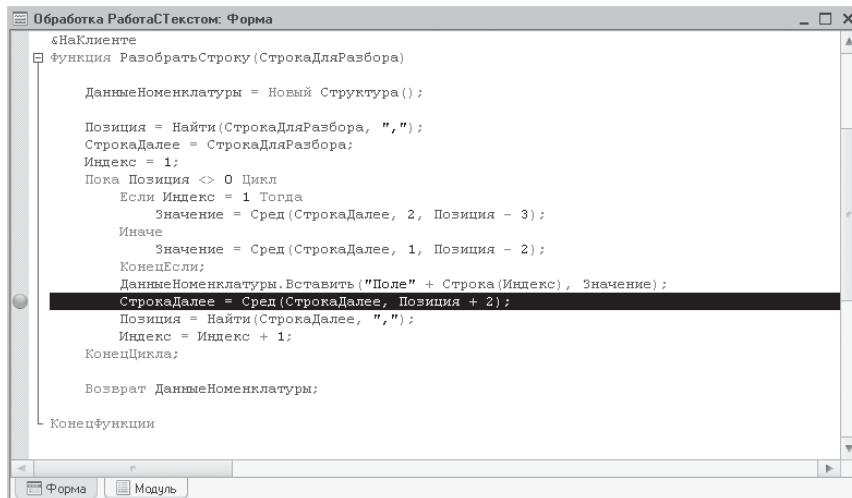


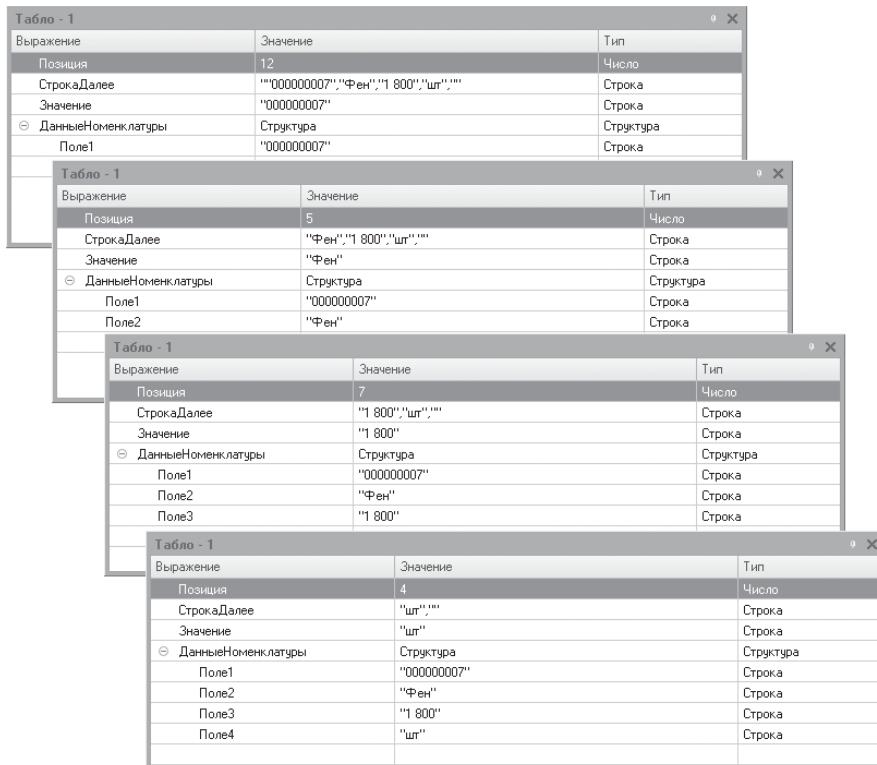
Рис. 162. Установка точки останова в цикле разбора строки

Нажмите кнопку Прочитать номенклатуру в форме демонстрационной обработки. Сработает точка останова, и управление будет передано в отладчик.

Чтобы в течение всего процесса отладки просматривать значения сразу всех выражений, участвующих в разборе строки, откройте окно Табло, нажав кнопку Табло (Ctrl + Alt + W) на панели инструментов конфигуратора. Затем

выделите мышью в модуле и перетащите в табло переменные СтрокаДалее, Позиция, Значение и структуру ДанныеНоменклатуры.

После этого, нажимая кнопку Продолжить отладку (F5) , вы можете останавливаться на каждом шаге цикла и смотреть, как заполняется каждое поле структуры данных (рис. 163).



Табло - 1		
Выражение	Значение	Тип
Позиция	12	Число
СтрокаДалее	"000000007","Фен","1 800","шг","",""	Строка
Значение	"000000007"	Строка
⊖ ДанныеНоменклатуры	Структура	Структура
Поле1	"000000007"	Строка

Табло - 1		
Выражение	Значение	Тип
Позиция	5	Число
СтрокаДалее	"Фен","1 800","шг","",""	Строка
Значение	"Фен"	Строка
⊖ ДанныеНоменклатуры	Структура	Структура
Поле1	"000000007"	Строка
Поле2	"Фен"	Строка

Табло - 1		
Выражение	Значение	Тип
Позиция	7	Число
СтрокаДалее	"1 800","шг","",""	Строка
Значение	"1 800"	Строка
⊖ ДанныеНоменклатуры	Структура	Структура
Поле1	"000000007"	Строка
Поле2	"Фен"	Строка
Поле3	"1 800"	Строка

Табло - 1		
Выражение	Значение	Тип
Позиция	4	Число
СтрокаДалее	"шг","",""	Строка
Значение	"шг"	Строка
⊖ ДанныеНоменклатуры	Структура	Структура
Поле1	"000000007"	Строка
Поле2	"Фен"	Строка
Поле3	"1 800"	Строка
Поле4	"шг"	Строка

Рис. 163. Пошаговое выполнение отладки

Если вы заметили, что какое-то поле структуры заполнилось неправильно, вы можете разобраться в этом более детально. Заново запустить отладку, дойти до «подозрительной» итерации и пройти ее по шагам, то есть останавливаясь после каждого выполненного оператора.

При этом текущая строка, в которой выполнена остановка, будет помечена стрелкой . Изначально она совпадает с точкой останова .

Например, остановившись в точке останова на второй итерации цикла, выполните отладку пошагово с помощью кнопки Шагнуть через (F10).

На приведенном ниже рисунке стрелкой отмечены изменения переменных после каждой выполненной строки функции (рис. 164).

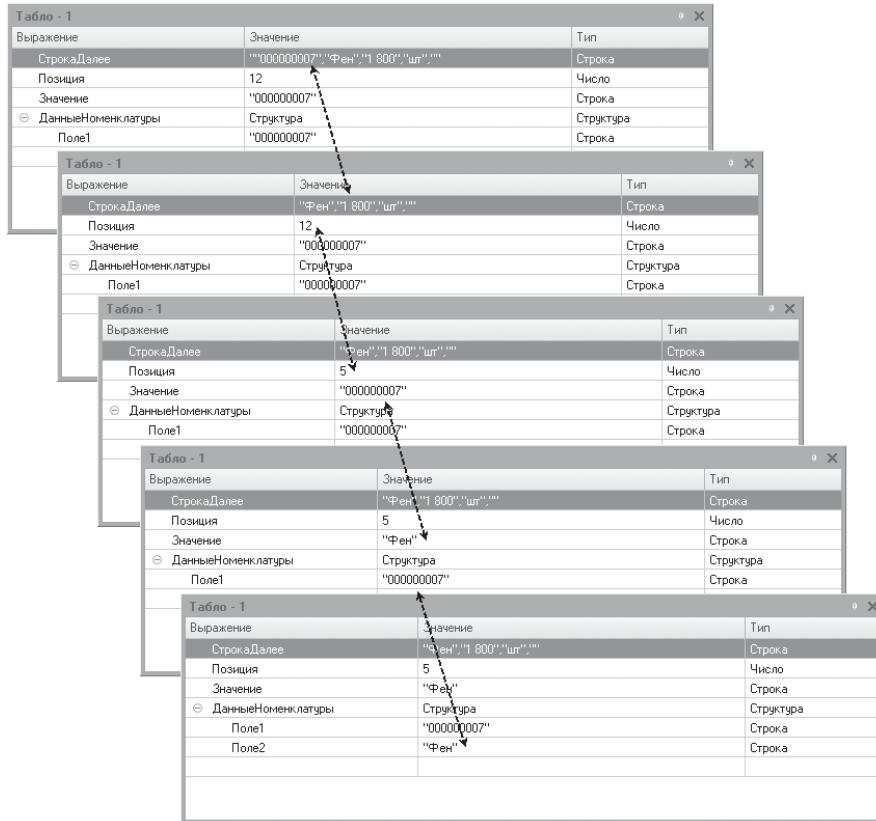


Рис. 164. Пошаговое выполнение отладки

Таким образом, с помощью пошаговой отладки вы можете поймать момент появления мелкой, но зловредной ошибки.

Пример 91. Узнать, откуда была вызвана процедура

Предположим, во время выполнения «1С:Предприятия» произошла ошибка, и вы перешли к месту ошибки в конфигуратор, внутрь какой-то процедуры. Посмотрели значения переменных и поняли, что данные, переданные в эту процедуру, уже были неправильные. В этом случае вам нужно посмотреть, откуда была вызвана эта процедура, чтобы перейти к месту вызова, и уже там найти ошибку.

Например, после возникновения ошибки мы перешли в конфигуратор в функцию ПолучитьАдрес(), которая возвращает адрес клиента. В эту функцию передается параметр СсылкаНаКлиента. При выполнении этой функции в строке, когда получается значение через точку от этого параметра, возникает ошибка «Значение не является значением объектного типа» (листинг 15).

Листинг 15. Функция «ПолучитьАдрес()»

```
&НаСервереБезКонтекста
Функция ПолучитьАдрес(СсылкаНаКлиента)

    Возврат СсылкаНаКлиента.Адрес;

КонецФункции
```

Вы посмотрели значение и тип параметра СсылкаНаКлиента в строке с ошибкой (выделенной в листинге 15 жирным шрифтом) в окне Выражение (Shift + F9). Вы видите, что параметр содержит строковое значение. Поэтому при получении от него значения через точку возникает ошибка, так как это не объект, а значение примитивного типа (рис. 165).

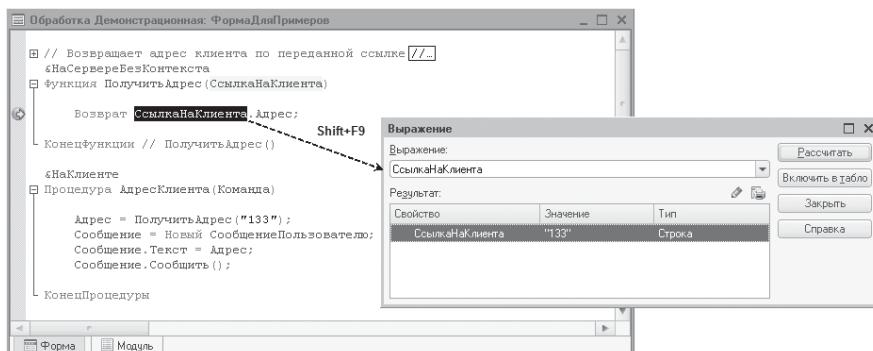


Рис. 165. Просмотр значения параметра при передаче в функцию

Возникает вопрос: откуда была вызвана эта функция? Почему в месте ее вызова параметр оказался не того типа, который нужен?

Чтобы увидеть, откуда была вызвана эта функция, нажмите кнопку Стек вызовов (**Ctrl + Alt + C**) на панели инструментов конфигуратора. В открывшемся окне Стек вызовов вы увидите последовательность вызовов процедур и функций, приведшую к строке с ошибкой (рис. 166).

Текущая функция **ПолучитьАдрес()**, которая отлаживается в данный момент, помечена стрелкой . Дважды щелкнув мышью по предыдущей строке, перейдите в процедуру **АдресКлиента()** к месту вызова этой функции (рис. 166).

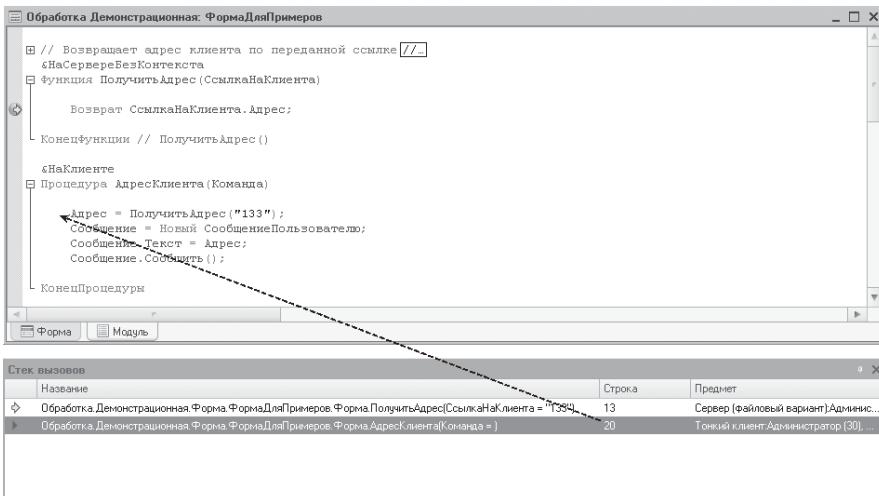


Рис. 166. Переход из окна стека вызовов к нужной процедуре

Проанализируйте место вызова функции **ПолучитьАдрес()** из процедуры **АдресКлиента()**, выделенное жирным шрифтом в листинге 16.

Листинг 16. Процедура «АдресКлиента()»

```

&НаКлиенте
Процедура АдресКлиента(Команда)

    Адрес = ПолучитьАдрес("133");
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = Адрес;
    Сообщение.Сообщить();

КонецПроцедуры
  
```

Теперь понятно, откуда берется строка («133») в функции, где возникает ошибка. Она передается в функцию в качестве фактического параметра, а на самом деле параметр должен иметь тип СправочникСсылка.Клиенты (как написано в описании функции).

Пример 92. Посмотреть результат выполнения запроса

Часто при отладке кода, использующего запросы, хочется убедиться в том, что из базы данных при помощи этих запросов читаются правильные данные.

Конечно, лучше всего отлаживать запросы в консоли запросов и затем переносить их в модуль. Но не всегда бывает легко «выдернуть» запрос из кода, особенно если он использует много параметров, которые вычисляются по ходу алгоритма. В таких случаях вы можете посмотреть результат запроса непосредственно в модуле. Удобнее всего вывести результат запроса в таблицу значений, остановившись после этого оператора и посмотреть в отладчике, что содержится в этой таблице значений.

Например, в обработке проведения документа перед тем, как формировать движения в регистрах накопления, вам нужно просмотреть и проверить те данные, из которых будут складываться движения документа.

Для этого найдите в процедуре обработки проведения строку, в которой запрос, с помощью которого формируются движения, выполняется методом Выполнить(). И после нее добавьте строку кода, в которой результат запроса выгружается в таблицу значений методом Выгрузить() (листинг 17).

Листинг 17. Выгрузка результатов выполнения запроса в таблицу значений

```
РезультатЗапроса = Запрос2.Выполнить();
T3 = РезультатЗапроса.Выгрузить();
ВыборкаДетальныеЗаписи = Результат.Выбрать();
```

Затем в коде модуля установите точку останова на следующую строку, чтобы таблица значений Т3 уже заполнилась результатами выполнения запроса. В нашем примере это строка ВыборкаДетальныеЗаписи = Результат.Выбрать(). После этого запустите «1С:Предприятие» в режиме отладки и проведите один из документов.

Когда исполнение кода будет остановлено, двойным щелчком мыши выделите переменную Т3 и нажмите кнопку Вычислить выражение (Shift + F9) на панели инструментов Отладка конфигурации. Откроется окно просмотра выражений.

Таблица значений является коллекцией, поэтому, чтобы просмотреть ее содержимое, выделите строку Т3 в окне Результат и нажмите кнопку Показать значения в отдельном окне (F2) над окном результата (рис. 167).

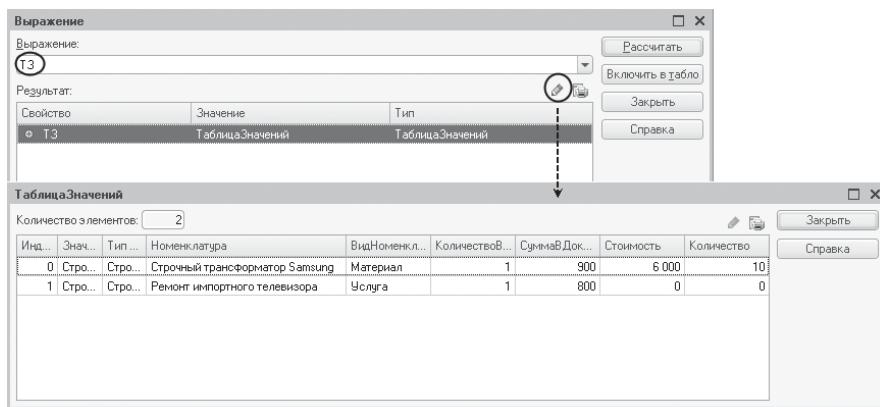


Рис. 167. Просмотр таблицы значений, содержащей результат запроса

В появившемся окне таблица значений, содержащая результат выполнения запроса, будет представлена в виде отдельных строк.

Пример 93. Выделить часть программы в отдельную процедуру

Часто бывает так, что вы пишете некоторый фрагмент процедуры, предназначающейся для разового использования. Например, для получения файлов и сохранения их в локальную файловую систему пользователя. Позднее, при доработке прикладного решения, оказывается, что этот же фрагмент алгоритма нужен вам еще раз. Или несколько раз. Для использования из этого же модуля или из других мест конфигурации.

Дублировать уже имеющуюся часть алгоритма из одной процедуры в другую неэффективно и даже опасно. Потому что потом, если понадобится модифицировать исходный фрагмент, легко забыть, в каких местах он используется.

В этом случае самое лучшее решение – выделить исходный фрагмент текста в отдельную процедуру или функцию и обращаться к ней из разных мест конфигурации.

Вы можете сделать это автоматически, выделив требуемый фрагмент текста и выполнив команду контекстного меню Рефакторинг – Выделить фрагмент.

При этом платформа анализирует дальнейшее использование выделенного фрагмента и на основе этого анализа автоматически выполняет следующие действия:

- Если этот фрагмент может выступать источником в операторе присваивания, то он будет выделен как функция. Также в функцию выделяется фрагмент, в котором инициализируется единственная переменная, использующаяся в оставшейся части родительского метода (из которого выделяется фрагмент). В остальных случаях фрагмент выделяется в процедуру.
- Если переменные выделяемого фрагмента используются в оставшейся части родительского метода, то они при необходимости предварительно инициализируются или объявляются в родительском методе и передаются в виде параметров в выделяемый метод.
- Если родительский метод предваряется какой-либо директивой компиляции, то выделяемый метод также будет предваряться точно такой же директивой компиляции.

В качестве примера рассмотрим процедуру, которая обходит список элементов справочника. От каждого элемента она берет хранимый в этом элементе файл и помещает его в массив. Затем все файлы, содержащиеся в массиве, она сохраняет в локальную файловую систему пользователя.

В начале процедуры, перед выполнением этих действий, формируется путь для выгрузки данных. А после выполнения этих действий анализируется, успешно ли прошло получение файлов. В случае неудачи выводится сообщение об ошибке.

Фрагмент текста, получающий файлы, выделен в листинге жирным шрифтом (листинг 18).

Листинг 18. Процедура «ПолучитьИСохранитьДанные()»

```
&НаКлиенте
Процедура ПолучитьИСохранитьДанные(СписокФайлов)

Каталог = КаталогВременныхФайлов();
ПутьВыгружаемыхФайлов = КаталогВременныхФайлов() + ?(Прав(Каталог, 1) = "\", "", "") + "Data\";

МассивФайлов = Новый Массив;
Для каждого ЭлементСписка Из СписокФайлов Цикл
    Файл = Новый Файл(Строка(ЭлементСписка.Значение));
    ПолучаемыйФайл = Новый ОписаниеПередаваемогоФайла;
    ПолучаемыйФайл.Имя = ЭлементСписка.Представление;
    ПолучаемыйФайл.Хранение =
```

```

ПолучитьНавигационнуюСсылку(ЭлементСписка.Значение, "Данные");
МассивФайлов.Добавить(ПолучаемыйФайл);
КонецЦикла;
ПолученныеФайлы = Новый Массив;
Результат = ПолучитьФайлы(МассивФайлов, ПолученныеФайлы, ПутьВыгружаемыхФайлов, Ложь);

Если НЕ Результат Тогда
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Ошибка получения файлов!";
    Сообщение.Сообщить();
КонецЕсли;

КонецПроцедуры

```

Выделите этот фрагмент, вызовите контекстное меню, раскройте подменю Рефакторинг и выполните команду Выделить фрагмент (рис. 168).

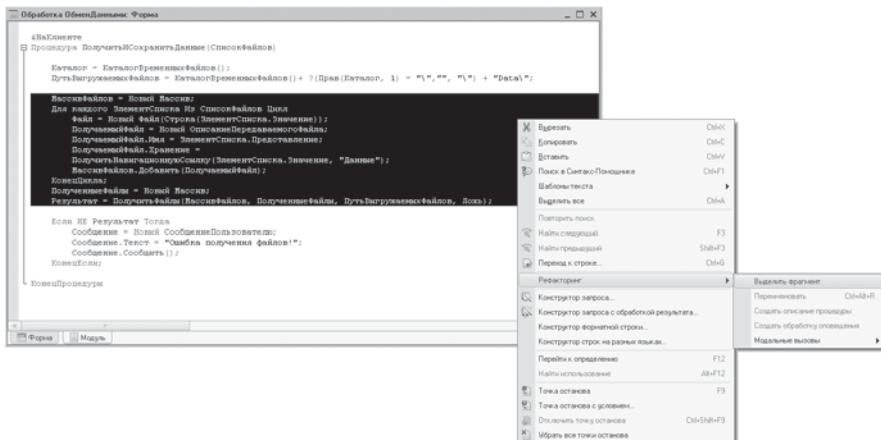


Рис. 168. Выделение фрагмента процедуры в отдельную функцию

Затем укажите имя создаваемой функции – ПолучитьФайлыДанных (рис. 169).

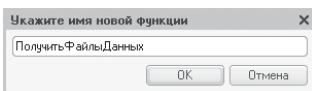


Рис. 169. Указание имени процедуры или функции,
в которую преобразуется выделенный фрагмент

Поскольку в оставшейся части метода анализируется результат получения файлов (переменная Результат), то выделенный фрагмент преобразуется в функцию, которая возвращает эту переменную.

Кроме того, в выделенном фрагменте используются список файлов с данными и путь для сохранения данных. Они будут переданы в функцию в виде параметров (листинг 19).

Листинг 19. Функция «ПолучитьФайлыДанных()»

```
&НаКлиенте
Функция ПолучитьФайлыДанных(Знач ПутьВыгружаемыхФайлов, Знач СписокФайлов)

Перем МассивФайлов, ПолучаемыйФайл, ПолученныеФайлы, Результат,
Файл, ЭлементСписка;

МассивФайлов = Новый Массив;
Для каждого ЭлементСписка Из СписокФайлов Цикл
    Файл = Новый Файл(Строка(ЭлементСписка.Значение));
    ПолучаемыйФайл = Новый ОписаниеПередаваемогоФайла;
    ПолучаемыйФайл.Имя = ЭлементСписка.Представление;
    ПолучаемыйФайл.Хранение =
        ПолучитьНавигационнуюСсылку(ЭлементСписка.Значение, "Данные");
    МассивФайлов.Добавить(ПолучаемыйФайл);
КонецЦикла;
ПолученныеФайлы = Новый Массив;
Результат = ПолучитьФайлы(МассивФайлов, ПолученныеФайлы, ПутьВыгружаемыхФайлов, Ложь);
Возврат Результат;

КонецФункции
```

А в процедуре `ПолучитьИСохранитьДанные()` выделенный фрагмент заменится на вызов этой функции (листинг 20).

Листинг 20. Процедура «ПолучитьИСохранитьДанные()»

```
&НаКлиенте
Процедура ПолучитьИСохранитьДанные(СписокФайлов)

Каталог = КаталогВременныхФайлов();
ПутьВыгружаемыхФайлов = КаталогВременныхФайлов() + ?(Прав(Каталог, 1) = "\"", "") + "Data\";

Результат = ПолучитьФайлыДанных(ПутьВыгружаемыхФайлов, СписокФайлов);

Если НЕ Результат Тогда
    Сообщение = Новый СообщениеПользователю;
    Сообщение.Текст = "Ошибка получения файлов!";
    Сообщение.Сообщить();
КонецЕсли;

КонецПроцедуры
```

Заметьте, что вновь созданная функция помещена в тот же модуль, что и родительская процедура. Однако если созданная вами функция `ПолучитьИСохранитьДанные()` будет использоваться в нескольких местах конфигурации, то имеет смысл поместить ее в общий модуль и при необходимости вызывать оттуда.

Пример 94. Изменить имя переменной или процедуры

Предположим, вы дорабатываете алгоритм, который писали некоторое время назад. Несмотря на то, что вы автор этого алгоритма, вам может быть уже трудно в нем разобраться. Имена переменных вам уже «ничего не говорят», назначение вызываемых функций из их имени также понять сложно. В общем, путаница.

Чтобы внести ясность и доработать алгоритм, а также исключить возможность такой неприятной ситуации в будущем, вы хотите переименовать некоторые процедуры и переменные.

Переименовывать вручную долго и неудобно, а, главное, чревато ошибками. В результате ваших усовершенствований прикладное решение может вообще перестать работать.

Поэтому быстрее, проще и безопаснее сделать это с помощью рефакторинга. При этом вы сможете переименовать как локальные, так и экспортируемые переменные, процедуры и функции. Для этого вам понадобится выполнить команду Переименовать подменю Рефакторинг.

Например, вам нужно переименовать локальную переменную Стр в процедуре модуля формы (листинг 21).

Листинг 21. Процедура «МатериалыКоличествоПриИзменении»

```
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
    Стр = Элементы.Материалы.ТекущиеДанные;
    РаботаСДокументами.Рассчитать(Стр);
КонецПроцедуры
```

Видно, что имя придумывалось на скорую руку и ни о чем нам не говорит. Измените его на более содержательное имя.

Установите курсор на имя этой переменной, вызовите контекстное меню, раскройте подменю Рефакторинг и выполните команду Переименовать (Ctrl + Alt + R). Затем укажите новое имя переменной – СтрокаТЧ (рис. 170).

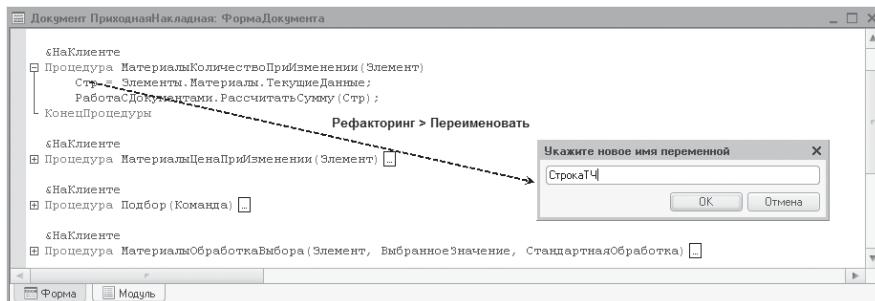


Рис. 170. Переименование локальной переменной модуля формы

В результате внутри локальной процедуры формы имя переменной будет заменено на новое во всех местах ее использования (листинг 22).

Листинг 22. Процедура «МатериалыКоличествоПриИзменении»

```
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
    СтроКатЧ = Элементы.Материалы.ТекущиеДанные;
    РаботаСДокументами.Рассчитать(СтроКатЧ);
КонецПроцедуры
```

Другой пример. Откройте экспортную процедуру Рассчитать() общего модуля. Она используется в нескольких документах для расчета суммы по строке табличной части (листинг 23).

Листинг 23. Процедура «РассчитатьСумму»

```
Процедура Рассчитать(СтроТабличнойЧасти) Экспорт
    СтроТабличнойЧасти.Сумма = СтроТабличнойЧасти.Количество * СтроТабличнойЧасти.Цена;
КонецПроцедуры
```

Имя процедуры тоже неудачное, так как не отражает назначение процедуры. Дайте процедуре более «говорящее» имя.

Установите курсор на имя этой процедуры в общем модуле и вызовите из контекстного меню команду Рефакторинг > Переименовать (Ctrl + Alt + R). Подтвердите, что вы хотите произвести глобальную замену во всех местах использования данной процедуры в конфигурации. Затем укажите новое имя процедуры РассчитатьСуммуСтроки (рис. 171).

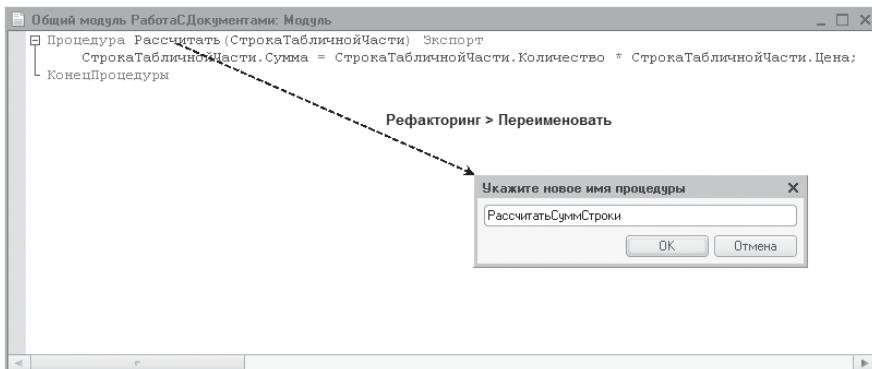


Рис. 171. Переименование экспортируемой процедуры общего модуля

В результате (при условии уникальности нового имени в рамках конфигурации) во всех местах использования этой процедуры в конфигурации имя процедуры будет заменено на новое (листинг 24).

Листинг 24. Процедура «МатериалыКоличествоПриИзменении»

```
&НаКлиенте
Процедура МатериалыКоличествоПриИзменении(Элемент)
    СтрокаТЧ = Элементы.Материалы.ТекущиеДанные;
    РаботаСДокументами.РассчитатьСуммыСтроки(СтрокаТЧ);
КонецПроцедуры
```

Третий пример. Если таким же образом вы будете переименовывать процедуры модуля формы, назначенные в качестве обработчика события самой формы или ее элементов, то платформа автоматически заменит ссылку на эти обработчики в соответствующих свойствах элементов формы и командах.

Установите курсор на имя процедуры – обработчика события в модуле формы и вызовите из контекстного меню команду Рефакторинг > Переименовать (Ctrl + Alt + R). Введите новое имя процедуры, нажмите OK.

Затем откройте палитру свойств, связанного с этой процедурой элемента формы и убедитесь, что имя процедуры-обработчика события здесь также изменилось (рис. 172).

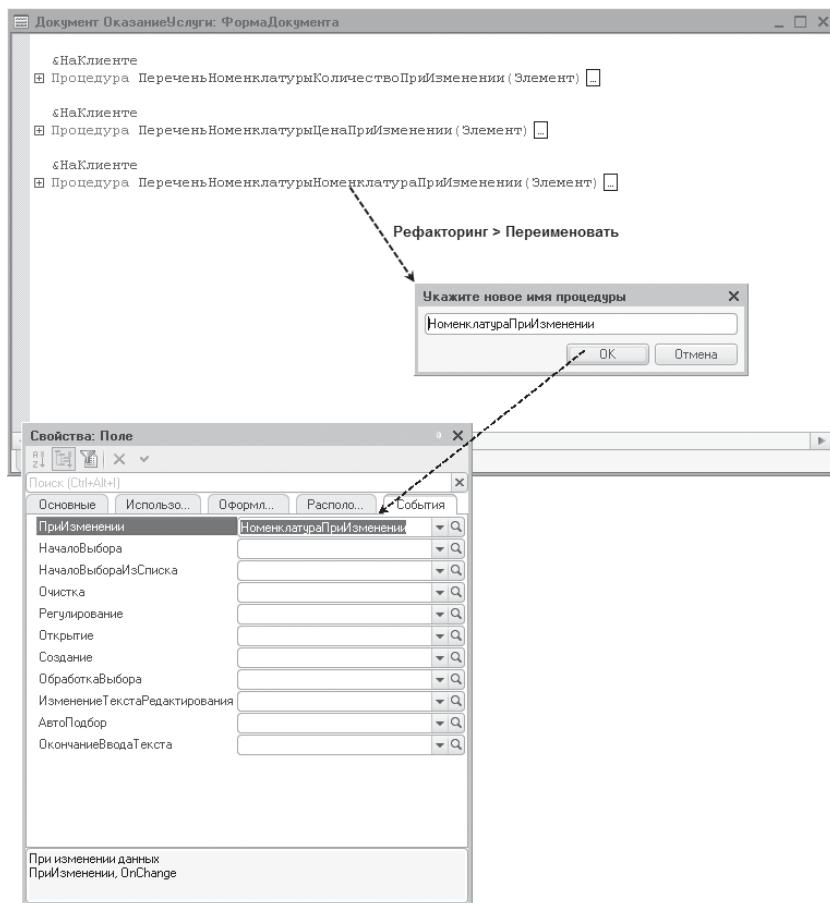


Рис. 172. Переименование процедуры – обработчика события
в модуле формы и в палитре свойств

В обратную сторону все работает точно так же. Когда выполняется переименование метода-обработчика в палитре свойств элемента формы или команды, то автоматически переименовываются все вызовы данного обработчика в модуле формы (рис. 173).

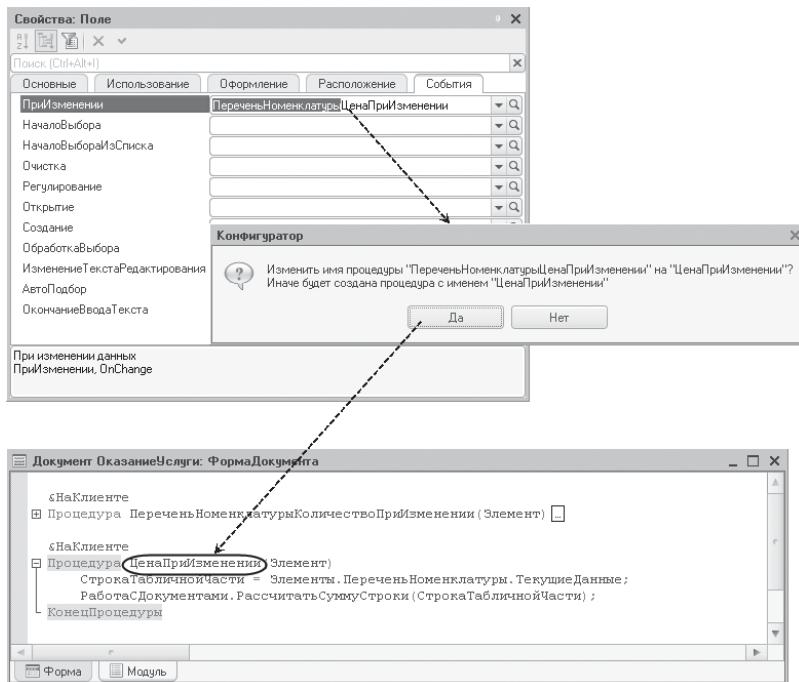


Рис. 173. Переименование процедуры – обработчика события в модуле формы и в палитре свойств

Пример 95. Создать описание процедуры

Предположим, вы дорабатываете функцию, написанную другими разработчиками, которые, к сожалению, не описали назначение и тип передаваемых в нее параметров по принятым правилам. Поэтому в месте вызова этой функции не появляется контекстная подсказка параметров с описанием их типа и назначения. Чтобы понять, что передавать в эту функцию, вам приходится смотреть пример ее вызова или разбираться с ее текстом. Это неудобно, но это можно легко и быстро исправить с помощью рефакторинга.

Для этого установите курсор в любое место функции и выполните команду контекстного меню **Рефакторинг > Создать описание функции** (рис. 174).

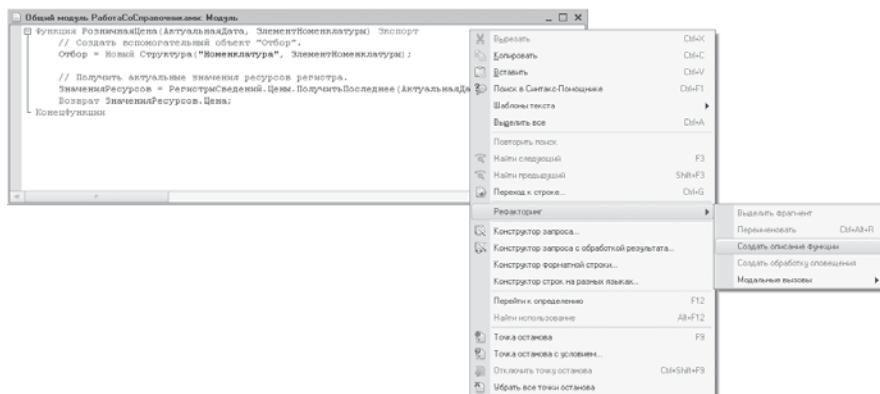


Рис. 174. Создание описания функции с помощью рефакторинга

После этого над функцией появится шаблон для ее описания (листинг 25).

Листинг 25. Шаблон описания функции с помощью рефакторинга

```
// Функция – Розничная цена
//
// Параметры:
// АктуальнаяДата      -      -
// ЭлементНоменклатуры -      -
// Возвращаемое значение:
// -
Функция РозничнаяЦена(АктуальнаяДата, ЭлементНоменклатуры) Экспорт
    // Создать вспомогательный объект "Отбор".
    Отбор = Новый Структура("Номенклатура", ЭлементНоменклатуры);

    // Получить актуальные значения ресурсов регистра.
    ЗначенияРесурсов = РегистрыСведений.Цены.ПолучитьПоследнее(АктуальнаяДата, Отбор);
    Возврат ЗначенияРесурсов.Цена;
КонецФункции
```

Остается только указать тип и назначение параметров функции и возвращаемого значения. Можно также уточнить назначение функции (по умолчанию это имя функции).

Пример 96. Создать обработку оповещения

Новые конфигурации мы рекомендуем разрабатывать в режиме без использования модальности. Особенно важной эта рекомендация является для приложений, которые будут работать на iPad, в режиме сервиса или под управлением веб-клиента.

Главной особенностью работы в этом режиме является то, что вместо модальных методов, открывающих модальные окна (`Вопрос()`, `Предупреждение()`, `ОткрытьЗначение()` и т.п.), вам нужно использовать их немодальные аналоги (`ПоказатьВопрос()`, `ПоказатьПредупреждение()`, `ПоказатьЗначение()` и т.п.).

Первым параметром в эти методы-аналоги передается объект `ОписаниеОповещения`. Он содержит описание экспортируемой процедуры – обработки оповещения. С этой процедуры будет продолжено исполнение кода после того, как пользователь закроет блокирующее окно.

Обработка оповещения может находиться в этом же модуле или в другом модуле, например, в общем модуле.

Таким образом получается, что в режиме без использования модальности «запрограммировать» вопрос или предупреждение не очень просто. Нужно создать заготовку процедуры обработки оповещения, сконструировать объект `ОписаниеОповещения`. И только после этого написать сам метод, который выведет вопрос или предупреждение.

Но, оказывается, есть способ автоматизировать эти действия. Для этого предназначена команда **Создать обработку оповещения** подменю **Рефакторинг**.

Например, перед тем как выполнить подбор номенклатуры в табличную часть документа, вам нужно задать вопрос пользователю.

Для этого в обработчике команды **Подбор** наберите имя метода `ПоказатьВопрос` и поставьте открывающую скобку. Затем установите курсор на имя метода и вызовите из контекстного меню команду **Рефакторинг > Создать обработку оповещения** (рис. 175).

Затем укажите имя процедуры-обработки оповещения (рис. 176).

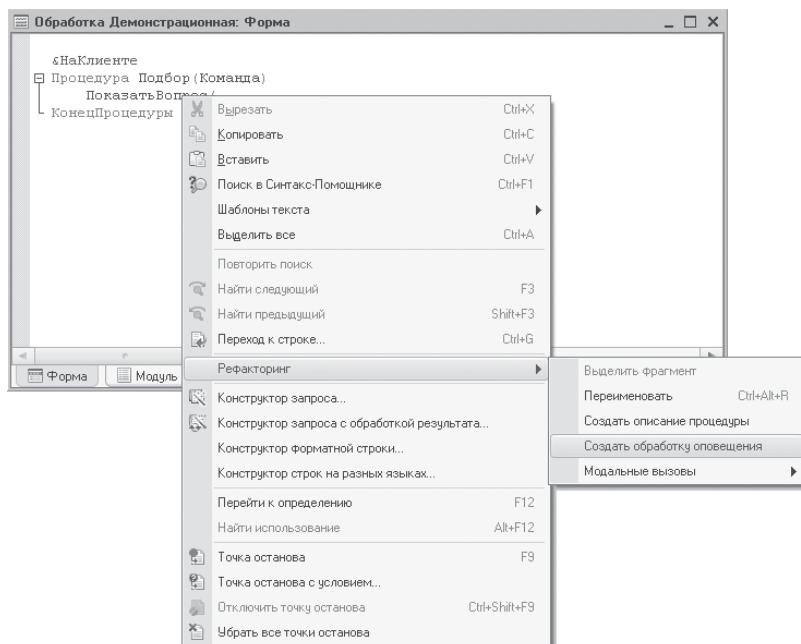


Рис. 175. Создание обработки оповещения с помощью рефакторинга

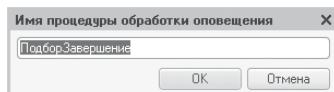


Рис. 176. Создание обработки оповещения с помощью рефакторинга

В результате в обработчике команды Подбор появится метод ПоказатьВопрос(). Первым параметром в нем будет конструктор объекта ОписаниеОповещения, который описывает процедуру ПодборЗавершение(), расположенную в этом же модуле и исполняющуюся после того, как пользователь ответит на вопрос. Шаблон процедуры-обработчика оповещения также будет создан в тексте модуля (листинг 26, 27).

Листинг 26. Обработчик команды «Подбор»

```
&НаКлиенте
Процедура Подбор(Команда)
    ПоказатьВопрос(Новый ОписаниеОповещения("ПодборЗавершение", ЭтотОбъект));
КонецПроцедуры
```

Листинг 27. Процедура «ПодборЗавершение()»

```
&НаКлиенте  
Процедура ПодборЗавершение(РезультатВопроса, ДополнительныеПараметры) Экспорт  
  
КонецПроцедуры
```

Заполните этот шаблон, например, следующим образом (листинг 28).

Листинг 28. Пример реализации, как задать вопрос пользователю
в немодальном режиме

```
&НаКлиенте  
Процедура Подбор(Команда)  
  
ПоказатьВопрос(Новый ОписаниеОповещения("ПодборЗавершение", ЭтотОбъект),  
"Подобрать номенклатуру в документ?", РежимДиалогаВопрос.ДаНет);
```

КонецПроцедуры

```
&НаКлиенте  
Процедура ПодборЗавершение(Результат, Параметры) Экспорт
```

```
Если Результат = КодВозвратаДиалога.Да Тогда  
    ПараметрыФормы = Новый Структура("МножественныйВыбор", Истина);  
    ОткрытьФорму("Справочник.Номенклатура.ФормаВыбора", ПараметрыФормы,  
        Элементы.Материалы);  
КонецЕсли;
```

КонецПроцедуры

В результате, когда пользователь нажмет кнопку Подбор, ему будет задан вопрос о необходимости подбора номенклатуры в документ. Когда пользователь ответит на вопрос, будет выполнена процедура обработки оповещения. И в случае положительного ответа будет открыта форма выбора элементов номенклатуры в режиме множественного выбора.

Пример 97. Создать строку на разных языках

Предположим, что в вашей конфигурации помимо русского языка добавлено еще два языка – английский и французский. И теперь вам нужно выводить, например, сообщения пользователю на том языке, который выбран для данного конкретного пользователя.

Для получения строки на языке текущего пользователя из набора строк на разных языках используется функция НСтр(). В эту функцию в виде строки передается набор строк на разных языках (вместе с кодами языков, определенных в конфигурации), разделенных точками с запятой.

Запоминать коды языков конфигурации, держать в голове синтаксис этой строки и писать ее вручную – долго и неудобно. Гораздо эффективнее формировать ее с помощью конструктора строк на разных языках.

Для этого в том месте, куда вам нужно поместить текст сообщения для пользователя, выполните пункт контекстного меню Конструктор строк на разных языках и подтвердите, что вы хотите создать новый набор строк (рис. 177).

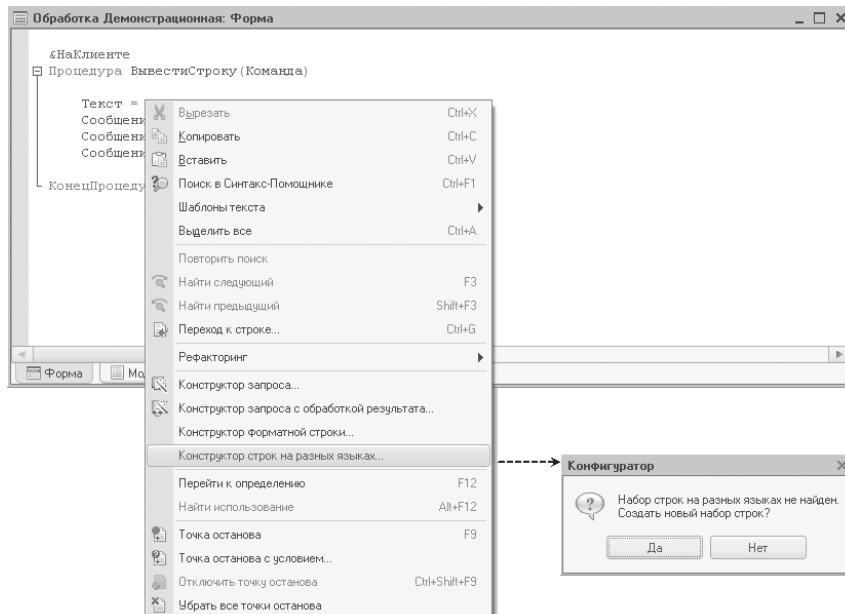


Рис. 177. Вызов конструктора строк на разных языках

В появившемся окне задайте строку, которую вы хотите показать пользователю (например, «Добрый день») на русском, английском и французском языках (рис. 178).

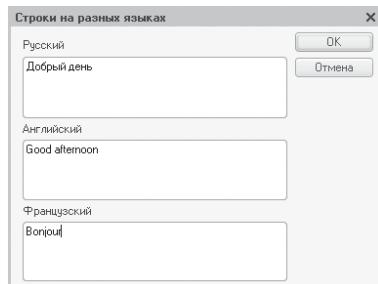


Рис. 178. Создание набора строк на разных языках

После нажатия ОК строка с текстом сообщения (он выделен в листинге жирным шрифтом) заполнится набором строк на трех языках. Вам остается только поставить точку с запятой в конце строки (листинг 29).

Листинг 29. Вывод сообщения пользователю на разных языках

```
&НаКлиенте  
Процедура СообщитьСтроку(Команда)
```

```
Текст = "ru = 'Добрый день!'; en = 'Good afternoon!'; fr = 'Bonjour'";  
Сообщение = Новый СообщениеПользователю;  
Сообщение.Текст = НСтр(Текст);  
Сообщение.Сообщить();
```

```
КонецПроцедуры
```

В результате выполнения приведенного выше кода для пользователя с установленным французским языком интерфейса будет выведено «Bonjour» (рис. 179).



Рис. 179. Сообщение пользователю на французском языке

В дальнейшем, установив курсор внутрь набора строк, вы можете снова открыть конструктор строк на разных языках из контекстного меню. При этом строки в окне конструктора будут соответствовать существующим значениям в исходной строке. Если изменить одну из строк, то существующая исходная строка будет заменена на новую.

Если же синтаксис исходной строки задан неверно или набор строк отсутствует, то конфигуратор предложит его создать (см. рис. 177).

Таким образом, с помощью конструктора строк на разных языках вы можете быстро, удобно и, главное, без ошибок задать новый набор строк или отредактировать существующий.

Пример 98. Изменить интерфейсные названия

Если в конфигурации определено более одного языка, вам понадобится указать для каждого объекта конфигурации строки на разных языках, которые будут отображаться в зависимости от того, какой язык выбран у текущего пользователя. В платформе есть средства, которые позволяют автоматизировать эту работу.

Предположим, в конфигурации определены два языка – русский и английский. Значит, наименования полей отчетов, форм, кнопок и т.п. вам нужно указывать теперь на каждом из этих двух языков (рис. 180).

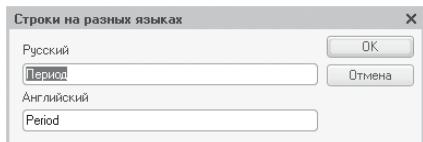


Рис. 180. Наименование поля отчета на разных языках

Допустим, теперь вы решили изменить англоязычное название какого-то поля отчета. Например, *Period*.

Во-первых, вы можете и не помнить, в каком отчете или форме это поле использовалось. Во-вторых, это слово может встречаться в конфигурации несколько раз, в разных полях. Отследить это вручную очень сложно.

В подобной ситуации механизм редактирования текстов интерфейса позволяет вам находить в конфигурации все строки на разных языках, группировать их по совпадающим значениям на каком-либо языке, копировать значения из одного языка в другой, а также выгружать значения строк из одной конфигурации и загружать их в другие.

Для перехода к редактированию выполните команду главного меню Правка > Редактирование текстов интерфейса. Сначала откроется окно настройки поиска, в котором ничего менять не нужно для решения нашей задачи (рис. 181).

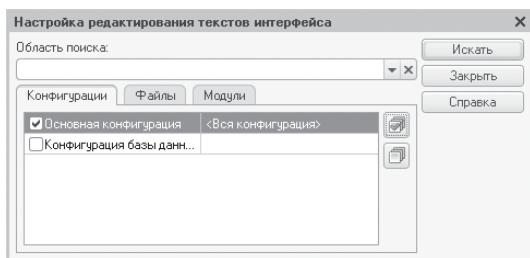


Рис. 181. Настройка редактирования текстов интерфейса

После нажатия кнопки Искать откроется окно, содержащее результаты поиска (рис. 182).

Редактирование текстов интерфейса			
Действия	Свернуть	Совпадающие на всех языках	
Расположение	Русский (ru)	Английский (eng)	ru1
Основная конфигурация Отчет.ОстаткиНоме...	Остатки номенклатуры по периодам		
«Вхождений: 5»	Отбор		
Основная конфигурация ОбработкаКонсоль...	Открыть файл отчетов		
Основная конфигурация ОбработкаКонсоль...	Открыть...		
«Вхождений: 2»	Отчет		
Основная конфигурация ОбработкаКонсоль...	Отчет нужно формировать		
Основная конфигурация Отчет.ОтчетПоПрод...	Отчет по продажам	Sales report	
«Вхождений: 2»	Отчет по системным блокам		
Основная конфигурация Отчет.ОстаткиНоме...	Оформление		
«Вхождений: 4»	Панель результатов		
«Вхождений: 3»	Параметры		
«Вхождений: 3»	Перечень номенклатуры		
«Вхождений: 5»	Период	Period	
Основная конфигурация Отчет.ЦеныКомп...	Период	Period	
Основная конфигурация Отчет.Поступлен...	Период	Period	
Основная конфигурация Отчет.Оказание...	Период	Period	
Основная конфигурация Отчет.СписокНо...	Период	Period	
Основная конфигурация Отчет.ОтчетПоП...	Период	Period	
Основная конфигурация Отчет.ОтчетПоПрод...	Период, день	Day	
Основная конфигурация Отчет.ОтчетПоПрод...	Период, месяц	Mounth	
«Вхождений: 4»	Печать		

Рис. 182. Редактирование текстов интерфейса

Все найденные строки располагаются в таблице, в строках которой показываются объекты конфигурации, содержащие строки на разных языках, а в колонках – все языки, заданные в конфигурации. На пересечении находятся значения строк.

Таблица может быть сгруппирована по значениям, совпадающим на всех языках или только на языке, по которому выполнена сортировка. Это облегчает чтение и редактирование таблицы. Например, на рис. 182 для параметра Период сразу для пяти отчетов добавлен англоязычный заголовок Period.

Значения строк на разных языках вы можете редактировать прямо в этой таблице. Значение, введенное для группы строк, автоматически дублируется в каждую строку, входящую в группировку. Таким образом вы можете заменить слово Period сразу во всех местах, где оно используется.

Хочется сказать, что данным режимом удобно пользоваться даже тогда, когда в конфигурации определен только один язык. Потому что этот режим позволяет автоматизировать процесс переименования сразу для всех вхождений.

Например, для реквизитов НачалоРаботы, ОкончаниеРаботы табличной части справочника Сотрудники и реквизитов ДатаНачала, ДатаОкончания табличной части документа НачисленияСотрудникам установлены свойства Формат и ФорматРедактирования в значение ДФ=dd/MM/yy. И, предположим,

в какой-то момент вам понадобилось поменять эти свойства сразу для всех реквизитов на $\text{ДФ}=dd.MM.yyyy$. То есть раньше дата у вас представлялась как 18/03/14, а теперь вы хотите, чтобы дата везде выглядела как 18.03.2014.

Менять форматы каждого реквизита по отдельности – долго, трудоемко, и к тому же велика вероятность, что вы забудете изменить какие-то реквизиты или где-нибудь ошибетесь.

А используя механизм редактирования текстов интерфейса, вы можете сделать это быстро и без ошибок (рис. 183).

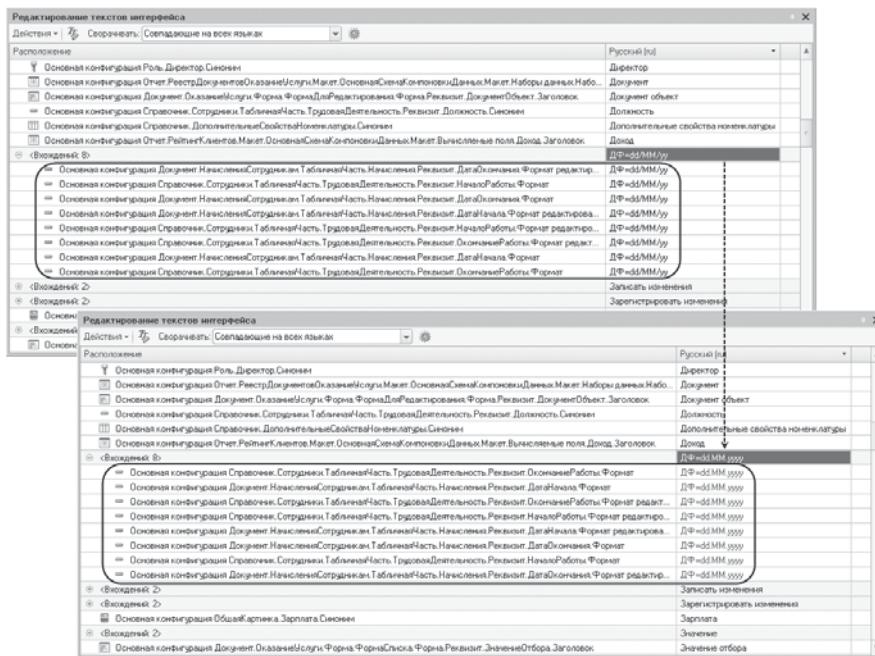


Рис. 183. Редактирование текстов интерфейса

Таким образом, хотя окно Редактирование текстов интерфейса предназначено для автоматизации процесса редактирования строк на разных языках, его эффективно использовать и для массового переименования полей отчетов, форм, кнопок и т. п. или изменения других совпадающих значений у объектов конфигурации на одном языке.

Пример 99. Выбирать поля в системе компоновки

Предположим, вам нужно задать настройки динамического списка: условное оформление, порядок или отбор. При выборе полей, по которым будут упорядочиваться, оформляться или отбираться данные в списке, удобно и быстро искать поля по первым буквам имени и сразу подставлять их из списка доступных полей.

Рассмотрим пример с условным оформлением динамического списка. Предположим, вы уже задали оформление (цвет шрифта, фона и т.п.) и условие для выделения полей списка.

Теперь вам нужно указать поля, которые будут оформлены при наступлении заданного условия.

В окне настроек динамического списка нажмите кнопку выбора в колонке **Оформляемые поля**. В открывшемся окне добавьте строку в список оформленяемых полей. А затем, вместо того чтобы нажимать кнопку выбора и вручную выбирать нужное поле из списка доступных полей, просто наберите первые буквы имени, и найденное (по мере ввода) соответствие само подставится в строку ввода (рис. 184).

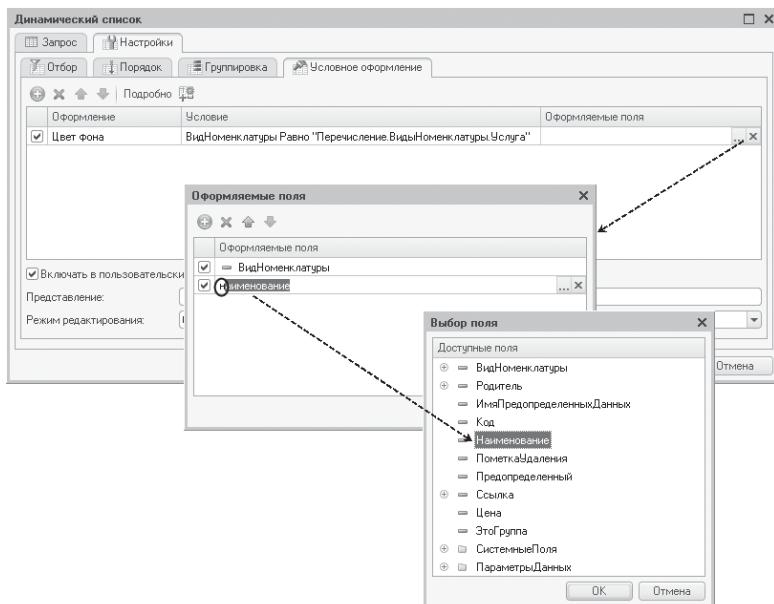


Рис. 184. Выбор полей в настройках динамического списка

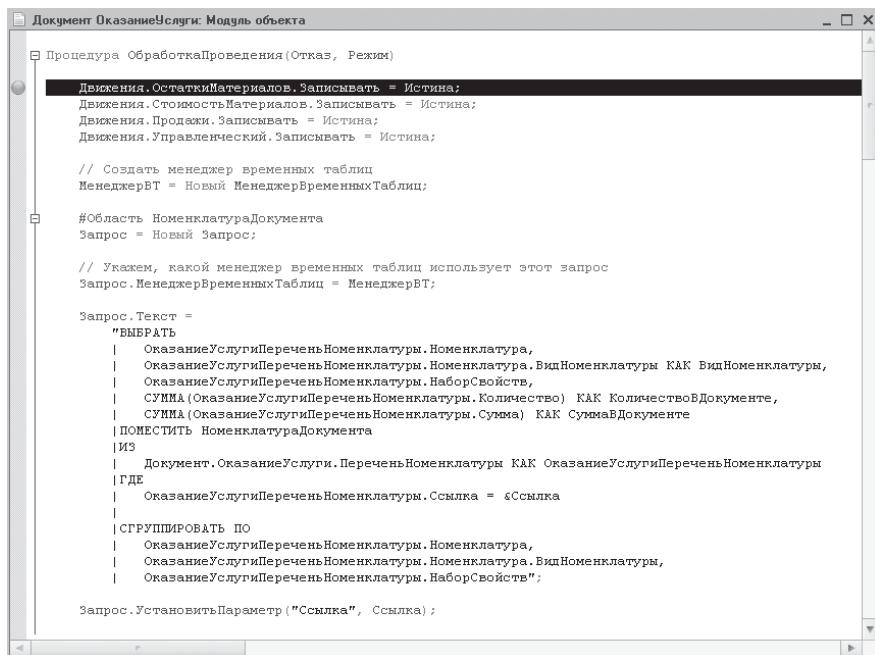
Подобным образом вы можете легко и просто выбирать поля в настройках отчетов и других объектах, использующих систему компоновки данных.

Пример 100. Оценить производительность программного кода

Часто бывает так, что вас не устраивает скорость работы написанного вами алгоритма. Например, проведение какого-то документа на большой базе выполняется слишком медленно. Вы хотите разобраться, в чем дело, и узнать, на выполнение каких операторов затрачивается наибольшее время.

Для этого вы можете выполнить замер производительности интересующего вас участка конфигурации, запущенной в режиме отладки.

Чтобы это сделать, откройте процедуру проведения документа и поставьте в начало этой процедуры точку останова (рис. 185).



```
Процедура ОбработкаПроведения(Отказ, Режим)
{
    Движения.ОстаткиМатериалов.Записывать = Истина;
    Движения.СтойкостьМатериалов.Записывать = Истина;
    Движения.Продажи.Записывать = Истина;
    Движения.Управленческий.Записывать = Истина;

    // Создать менеджер временных таблиц
    МенеджерВТ = Новый МенеджерВременныхТаблиц;

    #Область НоменклатураДокумента
    Запрос = Новый Запрос;

    // Укажем, какой менеджер временных таблиц использует этот запрос
    Запрос.МенеджерВременныхТаблиц = МенеджерВТ;

    Запрос.Текст =
        "ВЫБРАТЬ
        | ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
        | ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры КАК ВидНоменклатуры,
        | ОказаниеУслугиПереченьНоменклатуры.НаборСвойств,
        | СУММА (ОказаниеУслугиПереченьНоменклатуры.Количество) КАК КоличествоВДокументе,
        | СУММА (ОказаниеУслугиПереченьНоменклатуры.Сумма) КАК СуммаДокументе
        | ПОМЕСТИТЬ НоменклатуроДокумента
    |ИЗ
    | Документ.ОказаниеУслуги.ПереченьНоменклатуры КАК ОказаниеУслугиПереченьНоменклатуры
    |ГДЕ
    | ОказаниеУслугиПереченьНоменклатуры.Ссылка = «Ссылка
    |
    | СГРУППИРОВАТЬ ПО
    | ОказаниеУслугиПереченьНоменклатуры.Номенклатура,
    | ОказаниеУслугиПереченьНоменклатуры.Номенклатура.ВидНоменклатуры,
    | ОказаниеУслугиПереченьНоменклатуры.НаборСвойств";

    Запрос.УстановитьПараметр ("Ссылка", Ссылка);
}
```

Рис. 185. Точка останова в процедуре проведения документа

Запустите «1С:Предприятие» в режиме отладки и проведите интересующий вас документ. Сработает точка останова, и вы вернетесь в конфигуратор, в начало процедуры проведения.

После этого нажмите кнопку Замер производительности  на панели инструментов конфигуратора и затем продолжите отладку клавишей F5. Проведение документа продолжится. По его окончании снова перейдите в конфигуратор и нажмите кнопку Замер производительности еще раз.

После этого в конфигураторе откроется окно с результатами замера производительности кода конфигурации, который выполнялся между двумя нажатиями кнопки Замер производительности (рис. 186).

Модуль	Номер стр.	Строка	Кол.	Врем. (чистое)	%[Врем.] (чистое)
Документ.ОказаниеИспуги.Модуль...	140	Движение.Записать();	1	0,041200	41,90
Документ.ОказаниеИспуги.Модуль...	77	Движение.ОстаткиМатериалов.Запись();	1	0,012657	12,87
Документ.ОказаниеИспуги.Модуль...	76	Движение.СтоимостьМатериалов.Запис...	1	0,011977	12,18
Документ.ОказаниеИспуги.Модуль...	79	РезультатЗапроса = Запрос.Выполнить();	1	0,011069	11,26
Документ.ОказаниеИспуги.Модуль...	91	Если Выборка.Детальные.Записи.Номен...	6	0,010121	10,29
Документ.ОказаниеИспуги.Модуль...	38	РезультатЗапроса = Запрос.Выполнить();	1	0,003936	4,00
Документ.ОказаниеИспуги.Модуль...	4	Движение.ОстаткиМатериалов.Запись...	1	0,000776	0,79
Документ.ОказаниеИспуги.Модуль...	111	Движение = Движение.Управлениеский...	4	0,000475	0,48
Документ.ОказаниеИспуги.Модуль...	130	Движение = Движение.Продажи.Добави...	6	0,000424	0,43
Документ.ОказаниеИспуги.Модуль...	94	Движение.ОстаткиМатери...	4	0,000418	0,43
Документ.ОказаниеИспуги.Модуль...	112	Движение.СчетДт = ПланСчетов.Основ...	4	0,000377	0,38
Документ.ОказаниеИспуги.Модуль...	7	Движение.Управлениеский.Записывать ...	1	0,000358	0,36
Документ.ОказаниеИспуги.Модуль...	116	Движение.СубконтоДт.План.ВидеоКар...	4	0,000330	0,34
Документ.ОказаниеИспуги.Модуль...	5	Движение.СтоимостьМатериалов.Запис...	1	0,000315	0,32
Документ.ОказаниеИспуги.Модуль...	6	Движение.Продажи.Записывать = Истина;	1	0,000301	0,31
Документ.ОказаниеИспуги.Модуль...	125	Движение.СубконтоДт.План.ВидеоКар...	4	0,000202	0,21
Документ.ОказаниеИспуги.Модуль...	113	Движение.СчетКт = ПланСчетов.Основ...	4	0,000200	0,20
Документ.ОказаниеИспуги.Модуль...	119	Движение = Движение.Управлениеский...	4	0,000199	0,20
Документ.ОказаниеИспуги.Модуль...	99	Движение.Склад = Склад;	4	0,000198	0,20
Документ.ОказаниеИспуги.Модуль...	103	Движение = Движение.СтоимостьМатер...	4	0,000173	0,18
Документ.ОказаниеИспуги.Модуль...	120	Движение.СчетДт = ПланСчетов.Основ...	4	0,000142	0,14
Документ.ОказаниеИспуги.Модуль...	132	Движение.Номенклатура = Выборка.Дет...	6	0,000127	0,13
Документ.ОказаниеИспуги.Модуль...	97	Движение.Материал = Выборка.Детальн...	4	0,000123	0,13
Документ.ОказаниеИспуги.Модуль...	95	Движение.ВыдДвижения = ВыдДвижени...	4	0,000115	0,12
Документ.ОказаниеИспуги.Модуль...	85	Если Выборка.Детальные.Записи.Колич...	6	0,000111	0,11
Документ.ОказаниеИспуги.Модуль...	121	Движение.СчетКт = ПланСчетов.Основ...	4	0,000111	0,11
Документ.ОказаниеИспуги.Модуль...	83	Пока Выборка.Детальные.Записи.Следую...	7	0,000104	0,11
Документ.ОказаниеИспуги.Модуль...	135	Движение.Количество = Выборка.Деталь...	6	0,000093	0,09
Документ.ОказаниеИспуги.Модуль...	131	Движение.Период = Дата;	6	0,000089	0,09
Документ.ОказаниеИспуги.Модуль...	137	Движение.Стоимость = СтоимостьМатер...	6	0,000089	0,09
Документ.ОказаниеИспуги.Модуль...	136	Движение.Выричка = Выборка.Детальн...	6	0,000088	0,09
Документ.ОказаниеИспуги.Модуль...	100	Движение.Количество = Выборка.Деталь...	4	0,000082	0,08
Документ.ОказаниеИспуги.Модуль...	107	Движение.Стоимость = Выборка.Деталь...	4	0,000081	0,08
Документ.ОказаниеИспуги.Модуль...	88	СтоимостьМатерцала = Выборка.Деталь...	4	0,000077	0,08
Документ.ОказаниеИспуги.Модуль...	115	Движение.Сумма = Выборка.Детальн...	4	0,000075	0,08
Документ.ОказаниеИспуги.Модуль...	123	Движение.Сумма = СтоимостьМатериал...	4	0,000068	0,07
			1	0,041200	41,90

Для вызова процедур и функций включать время выполнения

Рис. 186. Результаты замера производительности

Чтобы быстро узнать, какая строка кода выполнялась дольше всего, вы можете просто щелкнуть мышью по заголовку колонки % (Врем.) и отсортировать результаты замера производительности в порядке убывания времени, затраченного на выполнение каждой строки кода. При этом сверху окажется строка, на выполнение которой затрачено наибольшее время.

Кстати, сортировка возможна по любой колонке таблицы результатов.

Проанализировав результаты замера производительности, вы можете сохранить эти результаты в файл с помощью команд Файл > Сохранить или Файл > Сохранить как (файл результатов имеет расширение *.pff).

Теперь вы можете попытаться как-то оптимизировать процедуру проведения документа, затем снова выполнить замер производительности, сравнить эти результаты с ранее сохраненными в файл результатами и т. д.

Также замер производительности может быть полезен при оптимизации клиент-серверного взаимодействия. При этом бывает нужно проанализировать серверные вызовы, которые выполняются платформой и/или производятся из клиентского кода.

Например, вы хотите узнать, сколько времени тратится на серверные вызовы при подборе пользователем номенклатуры в табличную часть документа. Поскольку это интерактивные действия пользователя, то для этого не надо останавливать приложение с помощью точки останова. Просто запустите конфигурацию в режиме отладки и нажмите кнопку Замер производительности на панели инструментов конфигуратора перед тем, как нажать кнопку Подбор, и после того, как окно подбора закроется.

В открывшемся окне с результатами замера производительности вы можете проанализировать, какой код выполнялся на клиенте (колонка Клиент), какой – на сервере (колонка Сервер), какие серверные вызовы выполнялись с клиента (колонка Обр. сервером), и какое время на это затрачивалось (рис. 187).

Модуль	Ном...	Строка	Кол.	Врем. чист...	%Врем...	Клиент	Сервер	Обр. сервером
Документ.ПриходноНаклад...	26	ОткрытьФорму[Справочник.Номенклатура.Фор...	2	0.995373	90.58	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	47	ПоказатьВопрос[Оповещение, "Добавить новен...	1	0.080588	7.33	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	82	НоваяСтрока = Объект Материалы.Добавить();	1	0.020044	1.82	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	31	КонецПроцедуры	2	0.001680	0.15	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	12	Если Значение Заполнено[Данные.ВидНоменкла...	18	0.000527	0.05	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	13	Представление = Данные.Наименование + " (" + ...	6	0.000422	0.04	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	50	НоваяСтрока = Объект Материалы.Добавить();	2	0.000212	0.02	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	11	СтандартнаяОбработка = Ложь;	18	0.000164	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	25	Параметры[Формы = Новый Структура]"Множест...	2	0.000110	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	18	КонецПроцедуры	18	0.000101	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	83	НоваяСтрока.Материал = ВыбранныйЭлемент;	1	0.000095	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	46	Оповещение = Новый ОписаниеОповещения["До...	1	0.000089	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	15	Представление = Данные.Наименование;	12	0.000088	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	51	НоваяСтрока.Материал = ВыбранныйЭлемент;	2	0.000084	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	16	КонецЕсли;	18	0.000065	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	81	Для Каждого ВыбранныйЭлемент Из СписокНом...	2	0.000058	0.01	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	49	Для Каждого ВыбранныйЭлемент Из Выбранное...	3	0.000049	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	4	Поля.Добавить["Наименование"];	1	0.000030	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	78	Если Результат = КодВозвратаДиалога Да Тогда	1	0.000029	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	44	Если ОтветПередДобавлением < Истина Тогда	2	0.000023	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	53	КонецЕсли;	2	0.000011	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	55	КонецПроцедуры	2	0.000011	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	3	СтандартнаяОбработка = Ложь;	1	0.000010	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	5	Поля.Добавить["ВидНоменклатуры"];	1	0.000009	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	52	КонецЦикла;	2	0.000007	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	79	ОтветПередДобавлением = Истина;	1	0.000006	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Справочник.Номенклатура.М...	7	КонецПроцедуры	1	0.000006	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	45	СтандартнаяОбработка = Ложь;	1	0.000006	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	88	КонецПроцедуры	1	0.000005	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	84	КонецЦикла;	1	0.000004	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Документ.ПриходноНаклад...	86	КонецЕсли;	1	0.000004	0.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
			2	0.995373	90.58			
Кол.	<input type="button" value="2"/>	Врем.	0.995373	%Врем.	90.58	<input checked="" type="checkbox"/> Клиент	<input type="checkbox"/> Сервер	<input type="checkbox"/> Для вызова процедур и функций включать время выполнения

Рис. 187. Результаты замера производительности

Пример 101. Найти неиспользуемые процедуры в коде конфигурации

Иногда получается так, что в модулях остаются «лишние», нигде не используемые процедуры.

Например, у элемента формы существовал обработчик события, а через какое-то время он стал не нужен. Вы удалили привязку события к этому обработчику из палитры свойств элемента, а саму процедуру-обработчик из модуля формы удалить по каким-то причинам забыли.

Или, например, в модуле формы существовала процедура для пересчета данных, потом вы решили использовать для этого процедуру общего модуля, а первоначальную процедуру из модуля формы удалить забыли.

Это, конечно, не приводит к ошибкам, но в конфигурации накапливается «мусор», который затрудняет ее сопровождение и снижает эффективность ее работы.

Но может быть и хуже, когда вы написали в модуле формы процедуру, чтобы она выполнялась при нажатии на кнопку, но забыли ее назначить обработчиком связанной с кнопкой команды. В результате данная процедура не будет исполняться в тот момент, когда вы этого ожидаете.

Таких ошибок можно избежать, если при создании обработчиков событий или команд автоматически связывать их с элементами формы (см. примеры «Создать кнопку» на стр. 115, «Создать обработчик события» на стр. 119). А при удалении этой привязки из палитры свойств элемента формы или команды также автоматически удалять сами процедуры из модуля формы (см. пример «Удалить обработчик события» на стр. 122).

Но в любом случае по окончании разработки конфигурации или ее некоторого этапа полезно проконтролировать себя путем поиска в модулях конфигурации неиспользуемых процедур и функций. Для этого выполните проверку конфигурации с помощью команды Конфигурация > Проверка конфигурации (рис. 188).

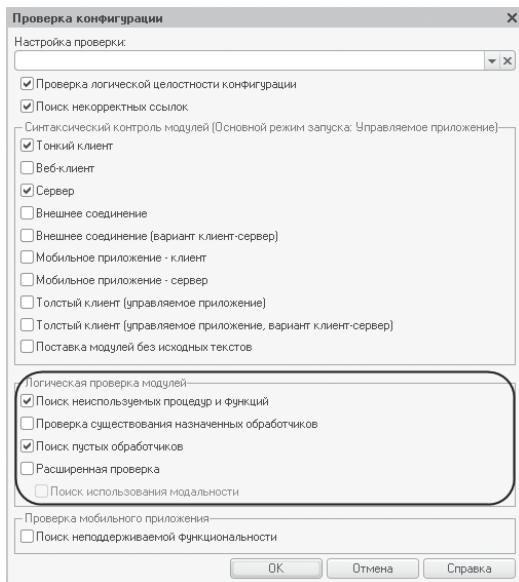


Рис. 188. Параметры проверки конфигурации

Опция Поиск неиспользуемых процедур и функций стандартно включена при проверке конфигурации в группе Логическая проверка модулей.

После нажатия ОК найденные ошибки будут показаны в окне служебных сообщений. Дважды щелкнув по строке с ошибкой, вы можете быстро перейти к соответствующему месту в модуле (рис. 189).

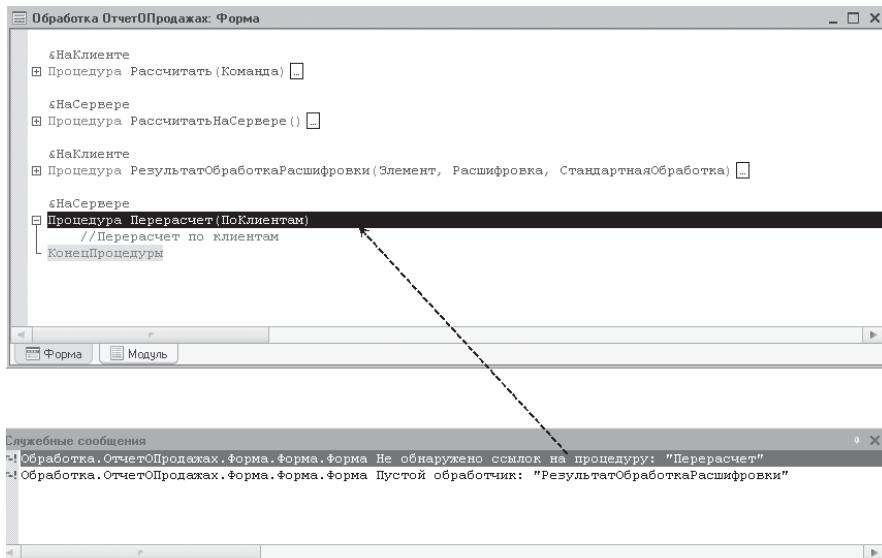


Рис. 189. Переход к ошибке, найденной при проверке конфигурации

В показанном примере в модуле формы обработки найдена процедура, которая не используется нигде в конфигурации. Это не приведет к ошибке, но текст процедуры, которая не нужна, будет скомпилирован на сервере, а это неэффективно. Кроме того, при сопровождении конфигурации у разработчика может возникнуть путаница и лишние вопросы.

© ООО «1С-Паблишинг», 2015

© Оформление. ООО «1С-Паблишинг», 2015

Все права защищены.

Материалы предназначены для личного индивидуального использования приобретателем.

Запрещено тиражирование, распространение материалов, предоставление доступа по сети к материалам без письменного разрешения правообладателей.

Разрешено копирование фрагментов программного кода для использования в разрабатываемых прикладных решениях.

Фирма «1С»

123056, Москва, а/я 64, Селезневская ул., 21.

Тел.: (495) 737-92-57, факс: (495) 681-44-07.

1c@1c.ru, <http://www.1c.ru/>

Издательство ООО «1С-Паблишинг»

127473, Москва, ул. Достоевского, 21/1, строение 1.

Тел.: (495) 681-02-21, факс: (495) 681-44-07.

publishing@1c.ru, <http://books.1c.ru>

Об опечатках просьба сообщать по адресу books.v8@1c.ru.