Улучшения в конфигураторе платформы 1С:Предприятие 8.3

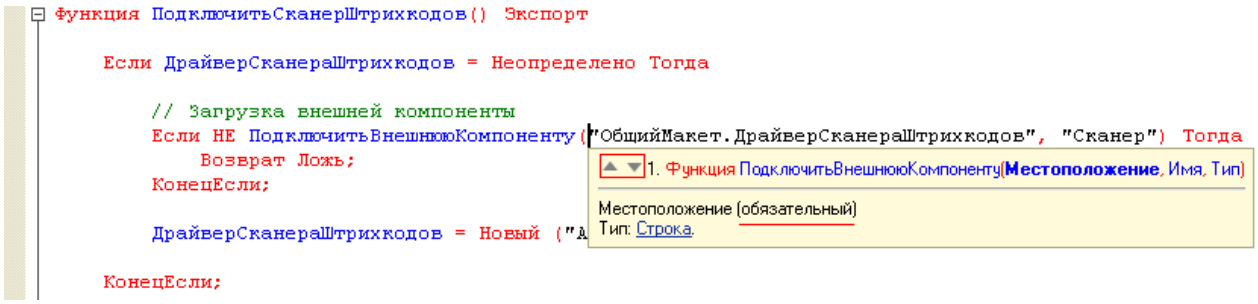
При выпуске новой версии платформы “1С:Предприятие” 8.3 разработчики добавили в неё несколько интересных и полезных нововведений, чтобы упростить ежедневный труд сотен разработчиков по всей стране.

Контекстная подсказка

Теперь при написании программного кода модуля в редакторе конфигуратора контекстная подсказка отображает не только допустимые в данном контексте имена переменных и процедур, но и параметры редактируемой в данный момент процедуры или функции.

Новая функциональность доступна как для встроенных процедур, так и для собственных процедур разработчика.

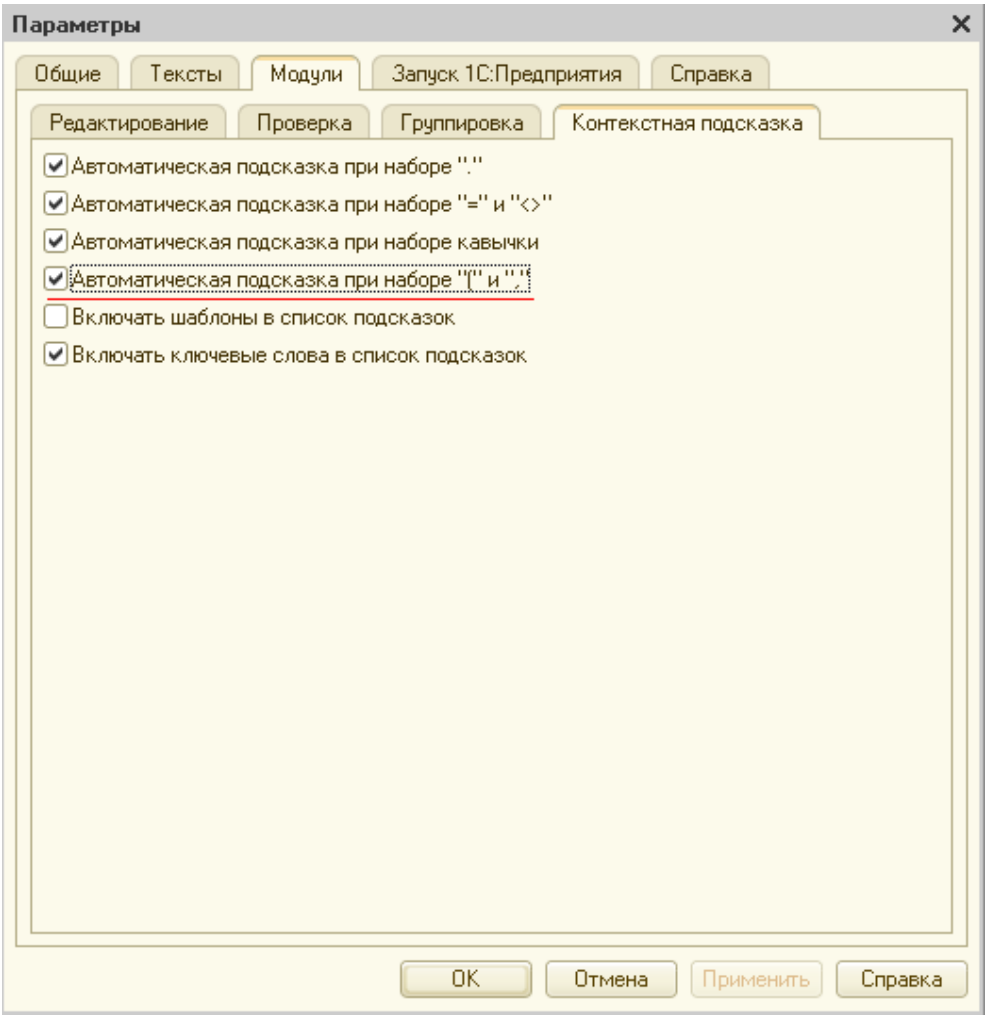
Подсказка со списком параметров выглядит следующим образом:



Параметр процедуры, который следует сейчас ввести, выделяется жирным шрифтом. Ниже под горизонтальной чертой располагается описание текущего параметра. Если он является обязательным, на этом акцентируется внимание при помощи текста в скобках.

При наличии нескольких вариантов синтаксиса встроенной процедуры в заголовке становятся доступны стрелки, предназначенные для переключения между этими вариантами.

Контекстная подсказка для параметров процедур и функций вызывается нажатием сочетания клавиш *Ctrl + Shift + Пробел*. Также ее можно вызвать автоматически при наборе символов "(“ и “, ”. Это поведение можно включить в диалоге параметров конфигуратора (пункт меню *Сервис - Параметры*, закладка *Модули - Контекстная подсказка*):



Следующей полезной особенностью новой контекстной подсказки является возможность отображать параметры пользовательских процедур и функций.

```
Если Не ВходящееПисьмо.Пустая() Тогда
    РаботаСПочтой.ЗаполнитьОтветНаПисьмо(ВходящееПисьмо, Объект, Содержимое);
КонецЕсли;
Адресаты = Параметры.Адресаты;
Если Адресаты <> Неопределено Тогда
    Запрос = Новый Запрос;
    Запрос.Текст = "ВЫБРАТЬ
                   |    Контрагенты.ЭлектроннаяПочта
                   |ИЗ
                   |    Справочник.Контрагенты КАК Контрагенты
```

Процедура ЗаполнитьОтветНаПисьмо(**ВходящееПисьмо**, ИсходящееПисьмо, Текст)

ВходящееПисьмо
Тип: СправочникСсылка.ВходящиеПисьма
- письмо, на которое следует ответить.

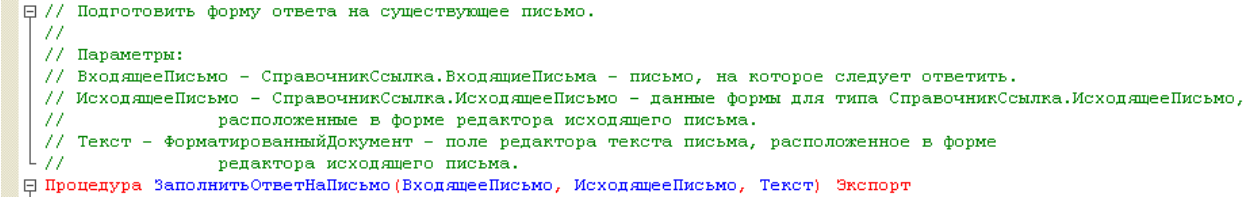
Напомним, что существует документ “*Система стандартов и методик разработки конфигураций для платформы 1С:Предприятие 8*”, в котором описаны рекомендации фирмы “1С” к разрабатываемому программному коду. В частности, есть рекомендации по оформлению комментария к заголовку процедуры.

Так, секция "*Параметры*" описывает параметры процедуры (функции). Если их нет, секция пропускается. Предваряется строкой "*Параметры:*", затем с новой строки размещаются описания всех параметров. Описание параметра начинается с новой строки, далее следуют имя параметра, затем дефис и список типов, далее – дефис и текстовое описание параметра.

Например:

```
// Подготовить форму ответа на существующее письмо.
//
// Параметры:
// ВходящееПисьмо - СправочникСсылка.ВходящиеПисьма - письмо, на которое следует ответить.
// ИсходящееПисьмо - СправочникСсылка.ИсходящееПисьмо - данные формы для типа СправочникСсылка.ИсходящееПисьмо,
//                   расположенные в форме редактора исходящего письма.
// Текст - ФорматированныйДокумент - поле редактора текста письма, расположенное в форме
//                   редактора исходящего письма.
Процедура ЗаполнитьОтветНаПисьмо(ВходящееПисьмо, ИсходящееПисьмо, Текст) Экспорт
```

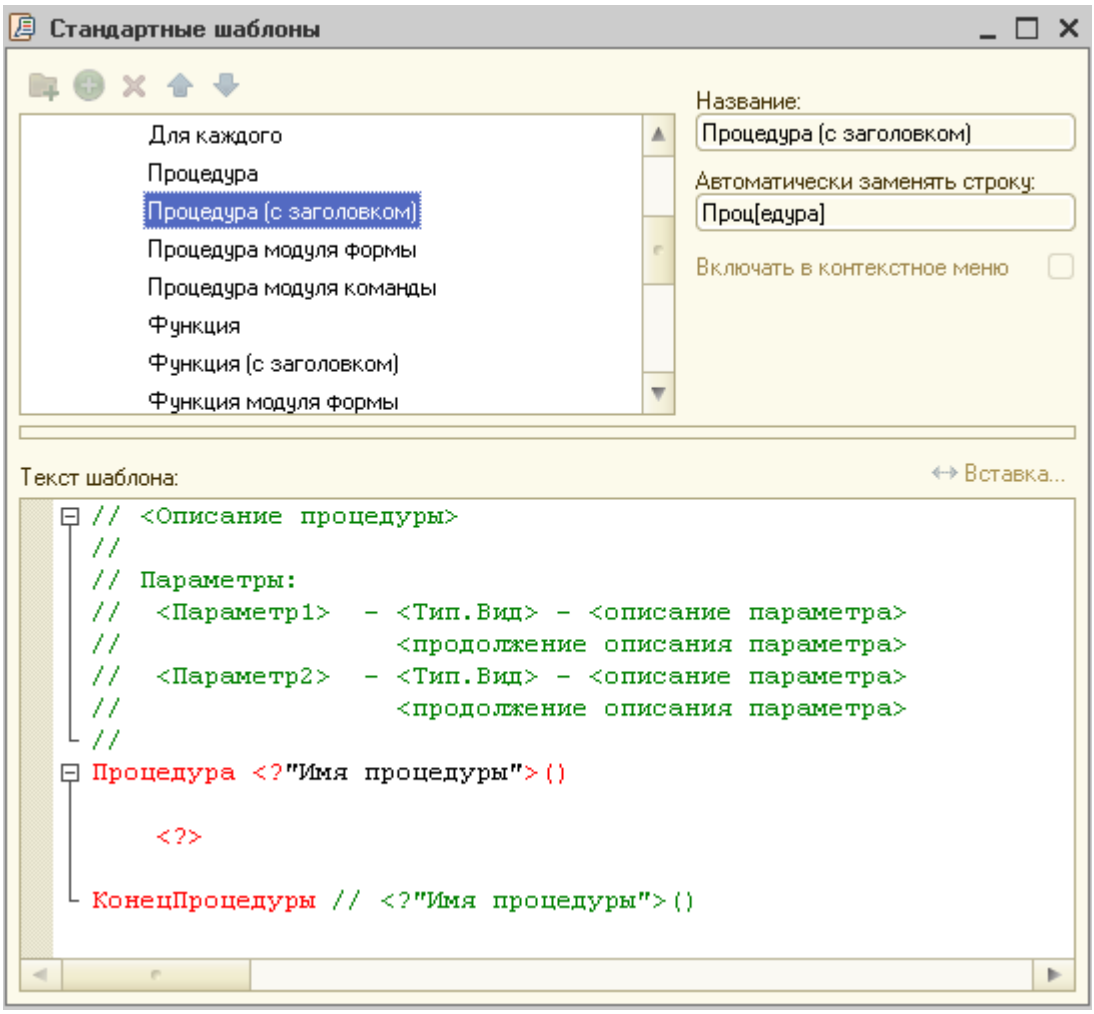
И конфигуратор анализирует комментарии, написанные по таким правилам, и использует их для отображения контекстной подсказки!



Чтобы избежать ручного написания комментария по приведенному формату, в платформе предусмотрены шаблоны текста, ознакомиться с которыми можно, нажав сочетание клавиш *Ctrl + Shift + T*.

Шаблон с наименованием "*Процедура (с заголовком)*" как раз и формирует правильный комментарий.

Чтобы этот шаблон сработал, достаточно набрать в редакторе символы "*Проц*", нажать *Ctrl+Q* и выбрать нужный шаблон из предлагаемого системой списка.



Группировка строк программного кода

Модули типовых решений на платформе “1С:Предприятие 8” достаточно объемные, содержат достаточно большое количество строк кода.

Для повышения удобства чтения и анализа программного кода были реализованы функции группировки условных и циклических операторов, а также процедур. Платформа 8.3 предоставляет еще одну возможность - сгруппировать произвольные строки модуля в одну группу по логическому принципу, а затем свернуть ее, чтобы она занимала меньше места на экране для повышения читабельности текста.

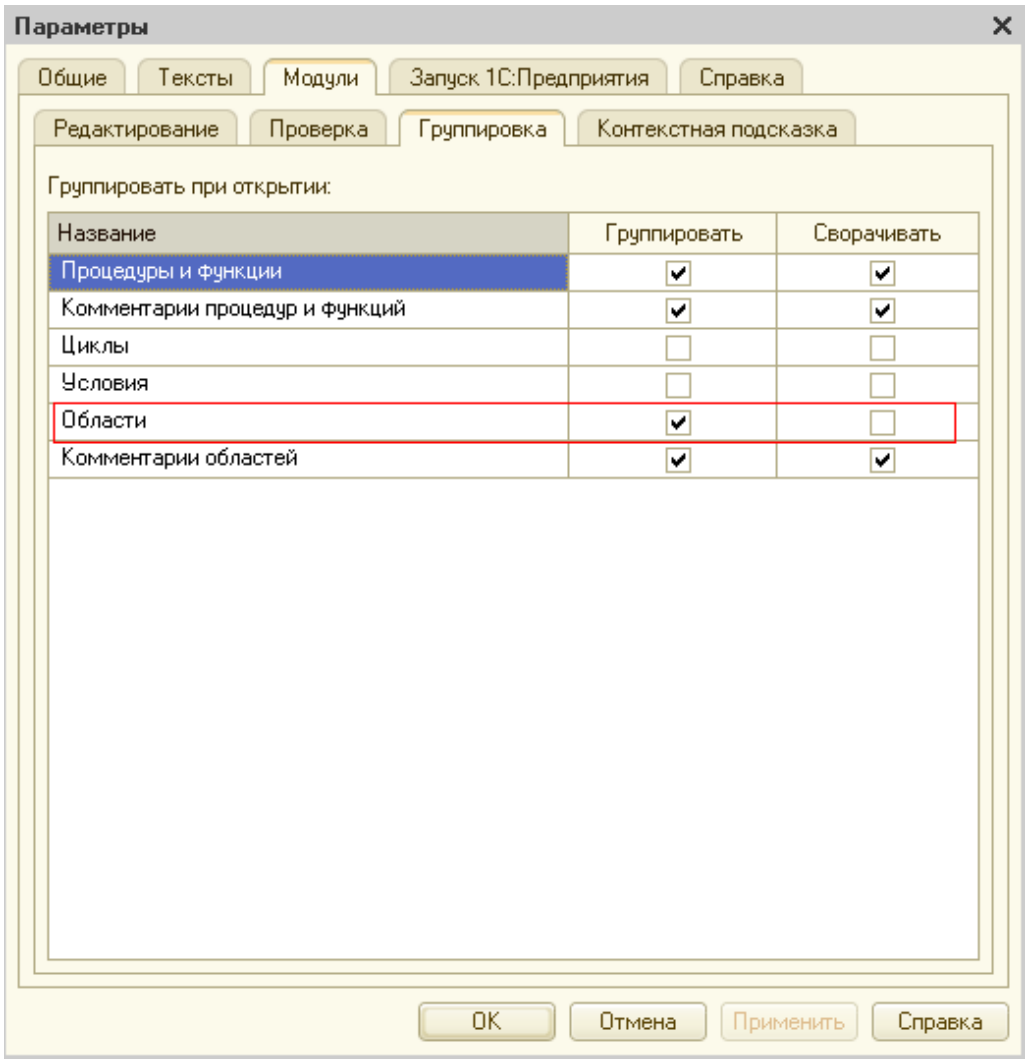
Для выделения области текста введены две новые инструкции препроцессора *#Область* и *#КонецОбласти*. Во время исполнения программного кода данные инструкции игнорируются. Они нужны только для обозначения сворачиваемых строк кода.

```
#Область А
Процедура ОбработкаЗаполнения(ДанныеЗаполнения, СтандартнаяОбработка)

    Если ТипЗнч(ДанныеЗаполнения) = Тип("СправочникСсылка.Контрагенты") Тогда

        ЗапросПоКонтрагенту = Новый Запрос("ВЫБРАТЬ
                                           |    Контрагенты.ЭтоГруппа
                                           |ИЗ
                                           |    Справочник.Контрагенты КАК Контрагенты
                                           |ГДЕ
                                           |    Контрагенты.Ссылка = &КонтрагентСсылка");
        ЗапросПоКонтрагенту.УстановитьПараметр("КонтрагентСсылка", ДанныеЗаполнения);
        Выборка = ЗапросПоКонтрагенту.Выполнить().Выбрать();
        Если Выборка.Следующий() И Выборка.ЭтоГруппа Тогда
            Возврат;
        КонецЕсли;

        Поставщик = ДанныеЗаполнения.Ссылка;
    КонецЕсли;

КонецПроцедуры
#КонецОбласти

Б
```

Нужно следить, чтобы группируемые области не пересекались между собой, потому что в таком случае они не будут сворачиваться на экране.

В конфигуратор добавлен шаблон текста для сокращения *#Обл*, который автоматически добавит в текст модуля инструкции по созданию новой области.

В диалоге параметров конфигуратора (пункт меню *Сервис* - *Параметры*, закладка *Модули* - *Группировка*) можно настроить группировку и сворачивание областей текста.



Выделение цветом конструкций

Теперь в редакторе текста на встроенном языке подсвечиваются цветом синтаксические конструкции, на которых в данный момент установлен курсор. Например, начало и конец процедуры (функции), условного оператора и оператора цикла:

```
Для каждого ТекСтрокаТовары Из Товары Цикл

    ОбластьТовары.Параметры.Заполнить(ТекСтрокаТовары);
    ТабличныйДокумент.Вывести(ОбластьТовары);

КонецЦикла;
```

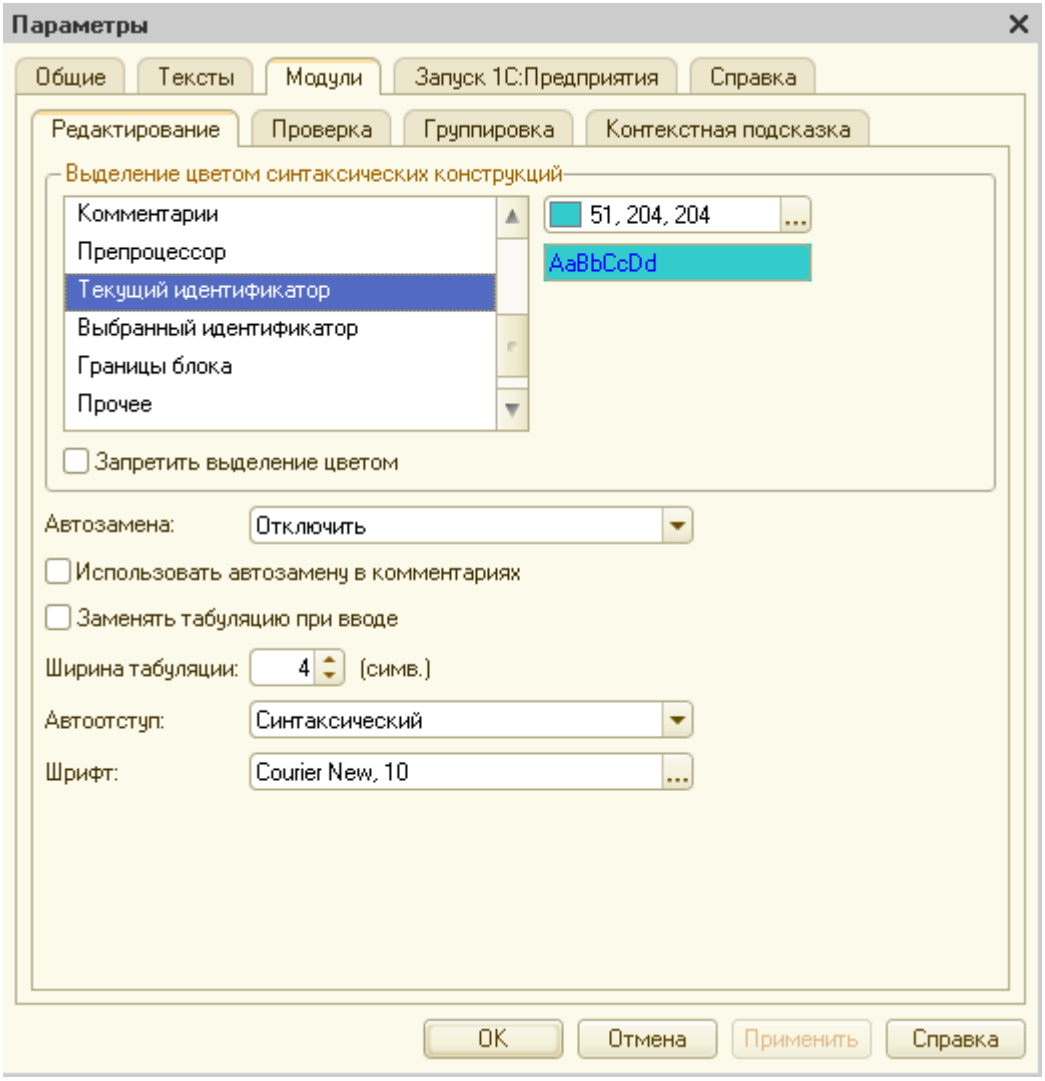
```
Процедура Пересчитать() Экспорт

    Для каждого ТекСтрокаТовары Из Товары Цикл

        ТекСтрокаТовары.Сумма = ТекСтрокаТовары.Количество * ТекСтрокаТовары.Цена;

    КонецЦикла;

КонецПроцедуры
```

Еще одним новшеством платформы является выделение цветом открывающихся и закрывающихся скобок. Это очень полезно при написании длинных выражений, когда синтаксический контроль сообщает об ошибке, и разработчику необходимо найти лишнюю или недостающую скобку.

```
ПутьФайла = СтроковыеФункцииКлиентСервер.РазложитьСтрокуВМассивПодстрок(СтрЗаменить(Обновление.ПутьКФайлуОбновления, "\", "/"), "/");
Если ПутьФайла.Количество() > 0 Тогда
    НоваяСтрока.ФайлОбновления = ПутьФайла[ПутьФайла.Количество() - 1];
КонецЕсли;
```

В диалоге параметров конфигуратора (пункт меню *Сервис - Параметры*, закладка *Модули - Редактирование*) можно настроить выделение цветом еще нескольких полезных конструкций.

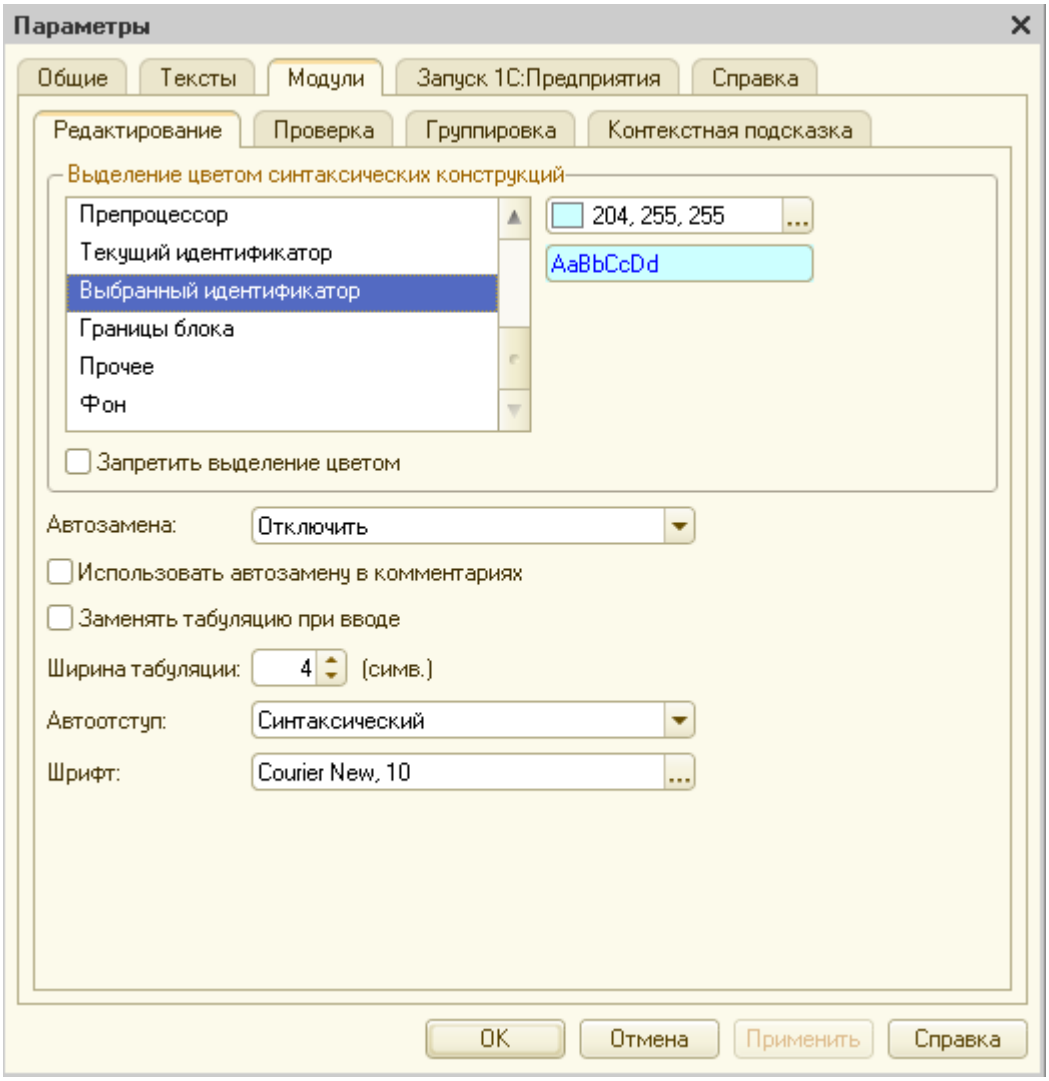
Если выбрать параметр “*Текущий идентификатор*” и назначить ему цвет, отличный от цвета фона редактирования (по умолчанию - белый), то при установке курсора на какой-либо идентификатор программного кода он сам выделяется выбранным цветом, а кроме того выделяются все такие же идентификаторы, встречающиеся в модуле, и строковые константы с заключенным в кавычки этим же идентификатором:




```
Функция ПодключитьСканерШтрихкодов() Экспорт

    Если ДрайверСканераШтрихкодов = Неопределено Тогда

        // Загрузка внешней компоненты
        Если НЕ ПодключитьВнешнююКомпоненту("ОбщийМакет.ДрайверСканераШтрихкодов", "Сканер") Тогда
            Возврат Ложь;
        КонецЕсли;

        ДрайверСканераШтрихкодов = Новый ("AddIn.Сканер.BarcodeReader");

    КонецЕсли;

    ИнформацияОСистеме = Новый СистемнаяИнформация;

    Если ИнформацияОСистеме.ТипПлатформы = ТипПлатформы.Windows_x86 ИЛИ ИнформацияОСистеме.ТипПлатформы = ТипПлатформы.Windows_x86_64 Тогда

        ТипОС = "Windows";

    ИначеЕсли ИнформацияОСистеме.ТипПлатформы = ТипПлатформы.Linux_x86 ИЛИ ИнформацияОСистеме.ТипПлатформы = ТипПлатформы.Linux_x86_64 Тогда

        ТипОС = "Linux";
```

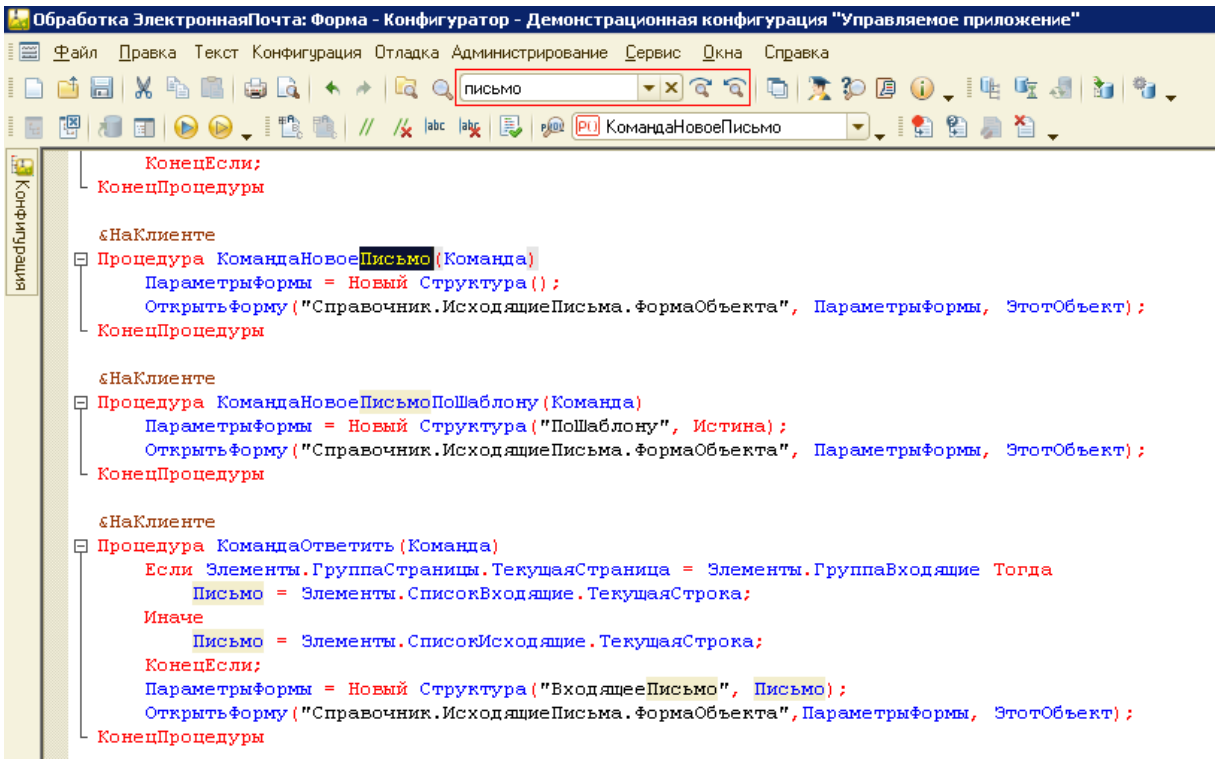
Также интерес представляет параметр “*Выбранный идентификатор*”. Если для него установлен цвет, не совпадающий с цветом фона редактирования, то при двойном

щелчке мышью по идентификатору будет подсвечен и он, и все совпадающие идентификаторы в тексте модуля.



```
// Формирование движения регистра накопления Взаиморасчеты.
Движения.Взаиморасчеты.Записывать = Истина;
Движение = Движения.Взаиморасчеты.Добавить();
Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
Движение.Период = Дата;
Движение.Контрагент = Покупатель;
Движение.Валюта = Валюта;

Если Валюта.Пустая() Тогда
    Движение.Сумма = Товары.Итог("Сумма");
Иначе

    Курс = РегистрыСведений.КурсыВалют.ПолучитьПоследнее(Дата, Новый Структура("Валюта", Валюта)).Курс;

    Если Курс = 0 Тогда
        Движение.Сумма = Товары.Итог("Сумма");
    Иначе
        Движение.Сумма = Товары.Итог("Сумма") / Курс;
    КонецЕсли;

КонецЕсли;
```

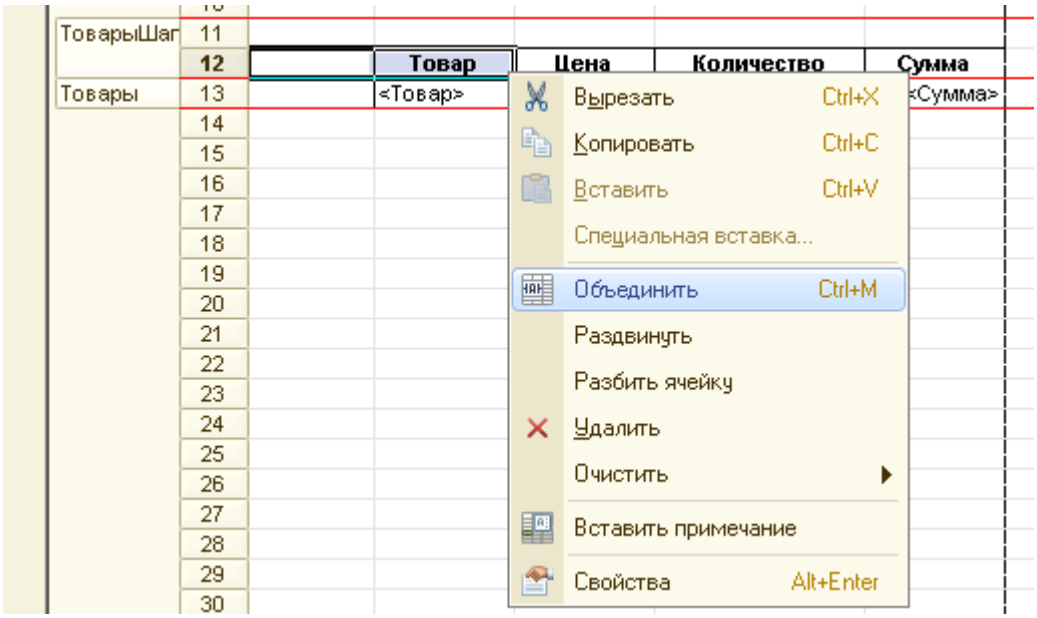
При выполнении поиска в тексте модуля при помощи строки поиска или после нажатия сочетания клавиш *Ctrl + F* найденное слово выделяется, а все такие же найденные слова подсвечиваются.



Объединение ячеек табличного документа

Ранее ячейки табличного документа можно было объединить только с помощью пункта меню или соответствующей кнопки командной панели.

Теперь появилось сочетание клавиш *Ctrl + M*, при нажатии которого и происходит объединение ячеек табличного документа. Также операция "*Объединить*" доступна в контекстном меню табличного документа.



Надеемся, что и в следующих релизах платформы "1С:Предприятие 8" разработчики будут уделять внимание повышению удобства работы с конфигуратором.

Новые возможности для разработчика в «1С:Предприятие 8.3.5»

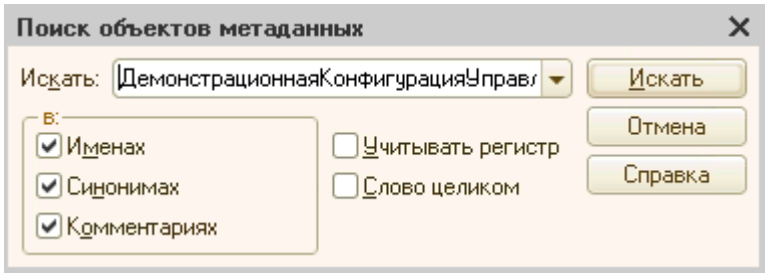
Поиск в конфигураторе

Пользоваться поиском при конфигурировании приходится постоянно. Пока конфигурация содержит относительно небольшое количество объектов метаданных, можно осуществлять поиск визуально – глазами, прокручивая дерево конфигурации.

Однако типовые конфигурации достаточно объемны, и при таком подходе поиск будет занимать длительное время.

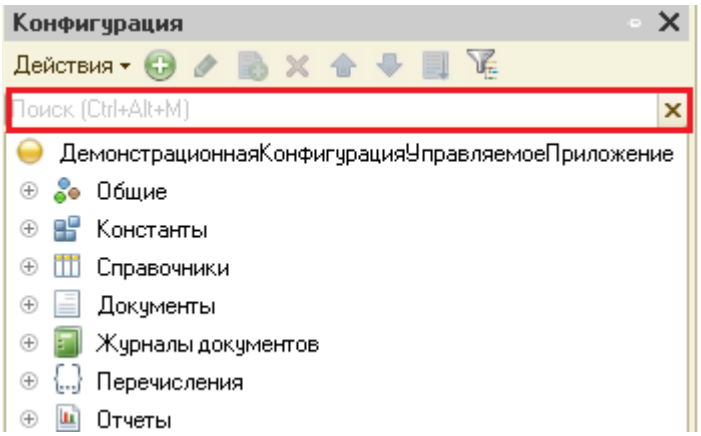
До выхода платформы 8.3.5 поиск по дереву метаданных можно было осуществить следующим образом:

- набирать с клавиатуры название объекта, при этом система будет искать по совпадению наименования с первой буквы названия, но только в развернутых строках дерева конфигурации;
- при помощи сочетания клавиш Ctrl+F открыть окно поиска:

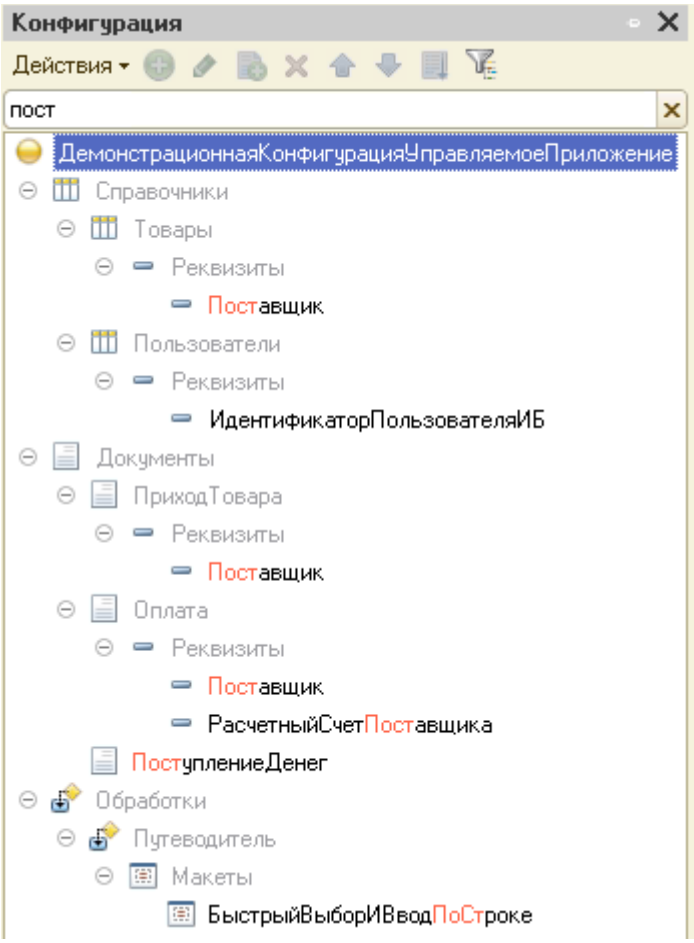


Найденные объекты будут выведены в окно *Результаты поиска*, из которого по двойному щелчку мышью можно перейти к нужному объекту метаданных в дереве конфигурации.

В платформе 8.3.5 появилось новое поле поиска, расположенное над деревом конфигурации:



Поиск выполняется по вхождению строки, анализируются свойствам объектов конфигурации *Имя*, *Синоним* и *Комментарий*. Причем дерево конфигурации фильтруется “на лету”: в нем остаются только объекты, удовлетворяющие введенному фильтру.



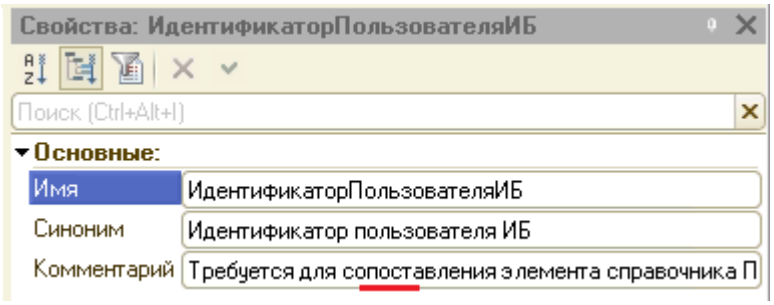
Рассмотрим, что обозначают цвета, которыми раскрашены объекты, оставшиеся в дереве после применения фильтра.

Если строка поиска была найдена, то имя такого объекта выделяется в дереве конфигурации черным цветом.

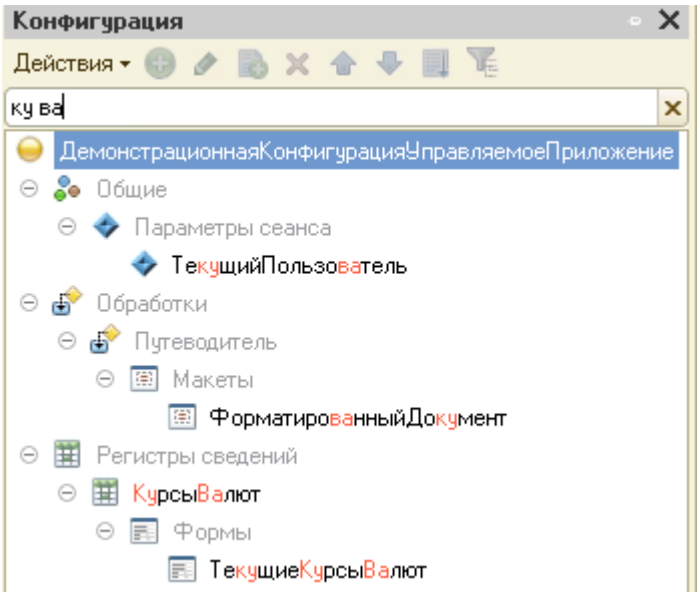
Если кроме того искомая строка присутствует в имени объекта (не в синониме, не в комментарии), то такие вхождения выделяются красным цветом.

Серым цветом выделяются объекты, сами не подходящие под введенный фильтр, но имеющие в своем составе подчиненные (дочерние) объекты, удовлетворяющие заданному фильтру.

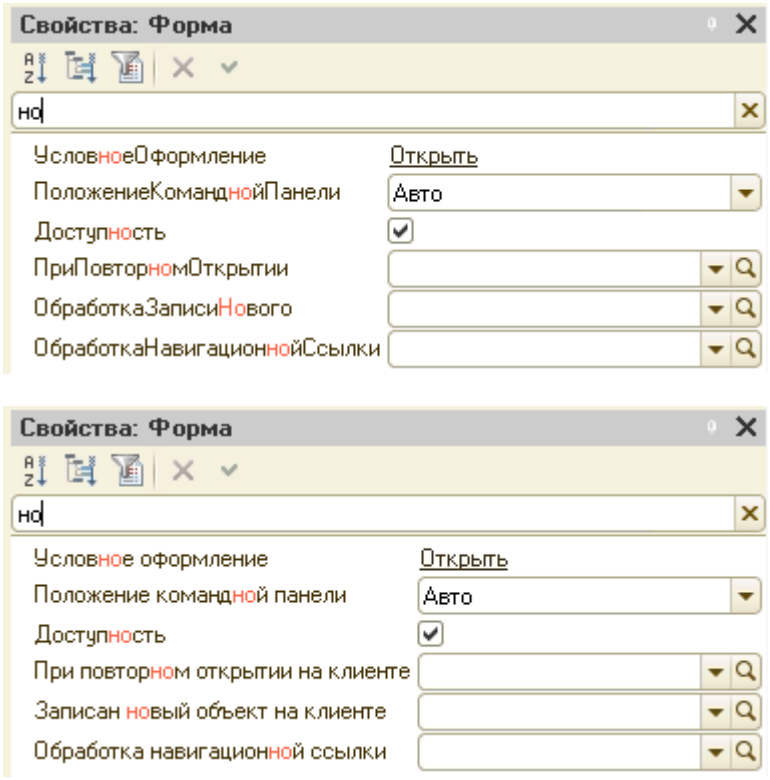
На приведенном выше рисунке реквизит *ИдентификаторПользователяИБ* справочника *Пользователи* отображается в дереве, т.к. его синоним содержит подстроку “пост”:



Допустимо вводить для поиска несколько подстрок, разделенных пробелами:

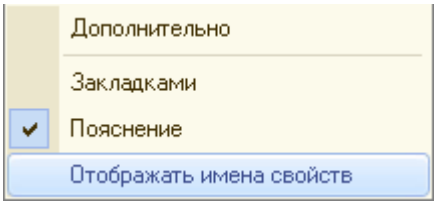


Аналогичная строка поиска появилось и у окна, содержащего набор свойств выделенного объекта (палитра свойств):

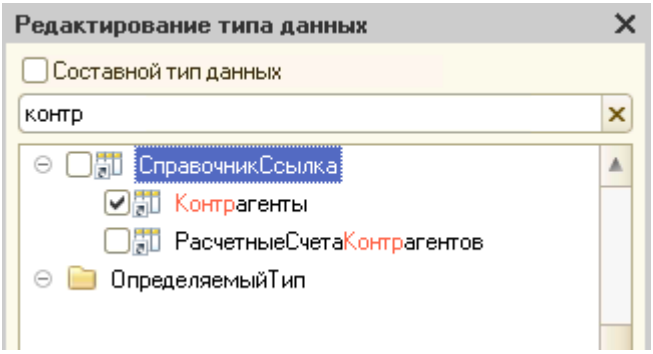


Найденные свойства будут выведены общим списком, без разбивки по категориям. Поиск будет осуществляться либо по именам свойств, либо по представлениям свойств (разница приведена на двух скриншотах выше).

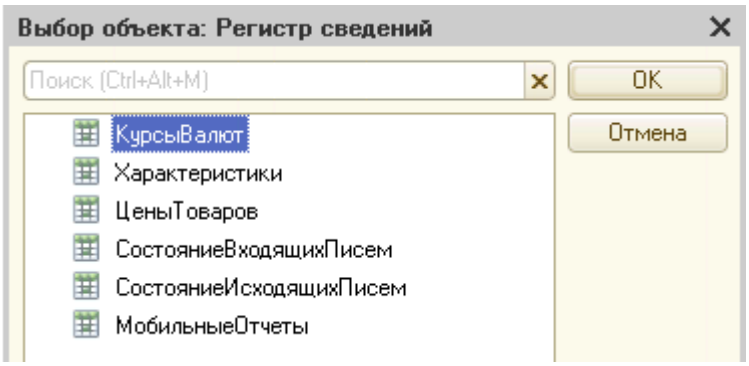
Переключиться между режимами имя/представление можно с помощью команды “Отображать имена свойств” контекстного меню:



Такая же строка поиска была добавлена в окне выбора типа данных:

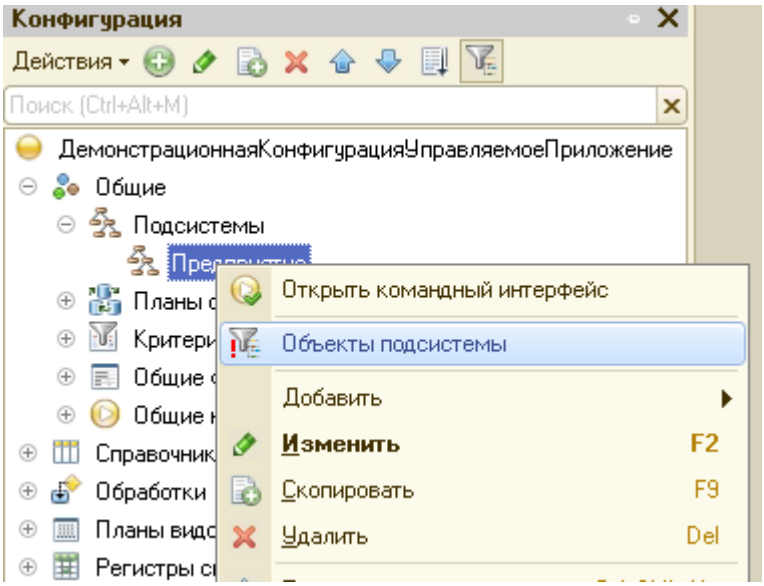


И в окно выбора объекта метаданных (например, выбора регистра сведений, который будет использоваться в качестве графика для регистра расчета):



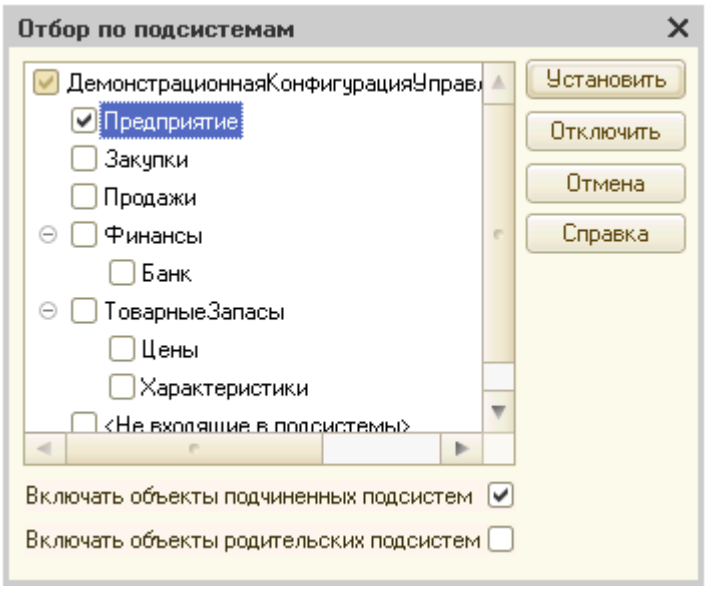
Быстрое отображение объектов подсистемы

Для быстрого отображения объектов, входящих в одну конкретную подсистему, в контекстном меню появился новый пункт “Объекты подсистемы”:



Напомним, как такого можно было добиться в предыдущих версиях платформы.

Нужно было открыть окно отбора по подсистемам, установить в нем галочку на требуемую подсистему, со всех остальных подсистем галочки снять:

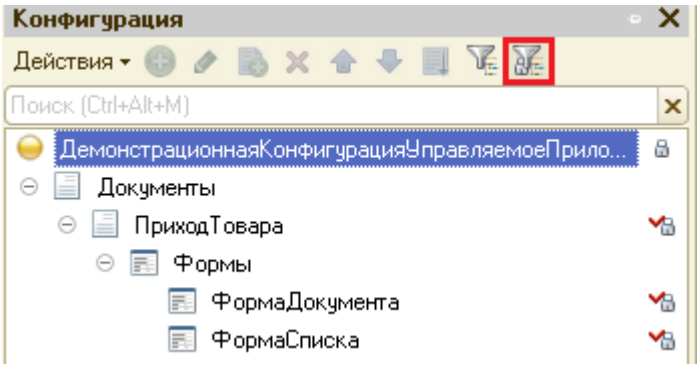


Теперь получить тот же самый результат можно быстрее. Кроме того, чаще всего используется и наиболее востребован отбор именно по одной подсистеме.

А, следовательно, это маленькое удобное новшество сэкономит время разработчика.

Быстрое отображение объектов, захваченных в хранилище

Если конфигурация подключена к хранилищу, то в командной панели над самим деревом конфигурации доступна кнопка “Захваченные объекты”:



Теперь фильтрация выполняется непосредственно в дереве конфигурации, не нужно открывать отдельное окно для работы с хранилищем, в нем устанавливать отборы на захваченные объекты.

Инструменты рефакторинга

Когда над конфигурацией работает группа из нескольких разработчиков, необходимо следить за понятностью кода, следованием общим стандартам.

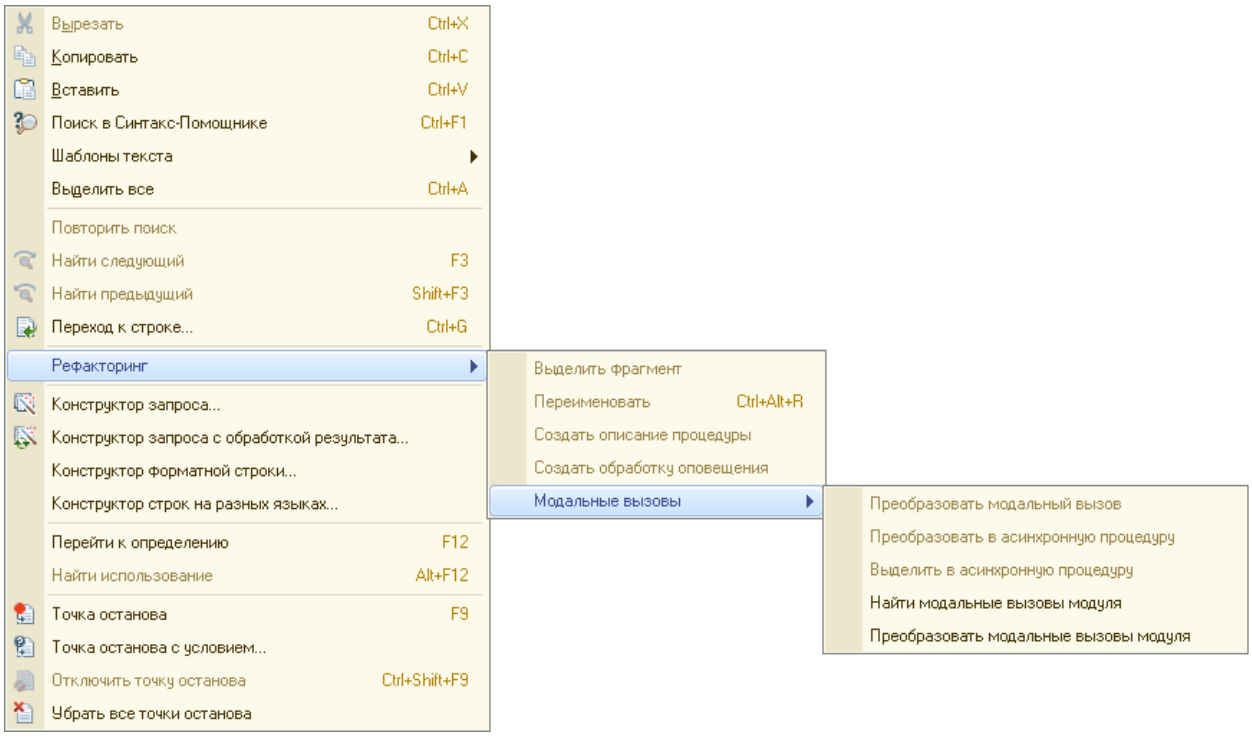
Контролировать это постоянно не всегда возможно, поэтому периодически проводятся работы по улучшению читаемости кода, пересмотру уже реализованных фрагментов.

Такие действия именуются рефакторингом кода. Это процесс изменения внутренней структуры программы, не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

Кроме этого разработчикам предстоит выполнить в своих конфигурациях работу по отказу от модальности - устранению модальных вызовов.

Поэтому в конфигураторе платформы 8.3.5 появились механизмы рефакторинга кода и инструменты работы с модальными вызовами.

Они доступны в контекстном меню текстового редактора конфигуратора в отдельном меню *Рефакторинг*.



Рассмотрим подробнее реализованные инструменты рефакторинга.

1. Выделить фрагмент

Эта команда преобразует выделенный участок кода в отдельную процедуру или функцию.

Если процедура, внутри которой расположен выделенный участок, содержит директиву компиляции (`&НаКлиенте`, `&НаСервере` и т.д.), то создаваемая процедура или функция будет иметь такую же директиву компиляции.

Если выделенный участок кода может быть расположен в правой части оператора присваивания, то будет создаваться функция. Рассмотрим пример. Пусть есть фрагмент кода:

```
&НаКлиенте
Процедура ТоварыТоварПриИзменении(Элемент)

    Стр = Элементы.Товары.ТекущиеДанные;
    Стр.Цена = ПолучитьЦенуТовара(Объект.Дата, Стр.Товар);
    Стр.Сумма = Стр.Количество * Стр.Цена;

КонецПроцедуры
```

Если применить команду “Выделить фрагмент” к выделенному участку кода, система сформирует следующий программный код (создаст новую функцию):

```
&НаКлиенте
Процедура ТоварыТоварПриИзменении(Элемент)

    Стр = Элементы.Товары.ТекущиеДанные;
    Стр.Цена = ПолучитьЦенуТовара(Объект.Дата, Стр.Товар);
    Стр.Сумма = РассчитатьСумму(Стр);

КонецПроцедуры

&НаКлиенте
Функция РассчитатьСумму(Знач Стр)

    Возврат Стр.Количество * Стр.Цена;

КонецФункции
```

Также функция будет создана, если в выделенном участке кода происходит присваивание одной переменной, которая используется ниже по коду. Например:

```
&НаКлиенте
Процедура ТоварыЦенаПриИзменении(Элемент)

    Стр = Элементы.Товары.ТекущиеДанные;
    Стр.Сумма = Стр.Количество * Стр.Цена;

КонецПроцедуры
```

Выделенный участок будет преобразован следующим образом:

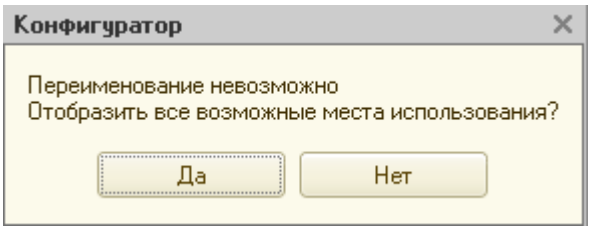
```
&НаКлиенте

Процедура ТоварыЦенаПриИзменении(Элемент)

    Стр = ТекущаяСтрокаТоваров();
    Стр.Сумма = Стр.Количество * Стр.Цена;

КонецПроцедуры

&НаКлиенте
Функция ТекущаяСтрокаТоваров()

    Перем Стр;

    Стр = Элементы.Товары.ТекущиеДанные
    Возврат Стр;

КонецФункции
```

2. Переименовать

Эта команда позволяет изменить имя переменной или процедуры (функции) во всех местах, где они фактически используются.

Если все вхождения переменной или метода определены однозначно, то система предложит указать новое имя и выполнит замену везде, где встречается этот идентификатор.

Если же все использования переменной или метода не могут быть идентифицированы однозначно, то система отображает вопрос и выводит места вхождения:



Рассмотрим ситуацию, когда система не сможет автоматически заменить имя процедуры. Пусть в модуле документа существует процедура:

```
Процедура Пересчитать() Экспорт

    Для каждого ТекСтрокаТовары Из Товары Цикл
     ТекСтрокаТовары.Сумма = ТекСтрокаТовары.Количество * ТекСтрокаТовары.Цена;
    КонецЦикла;

КонецПроцедуры
```

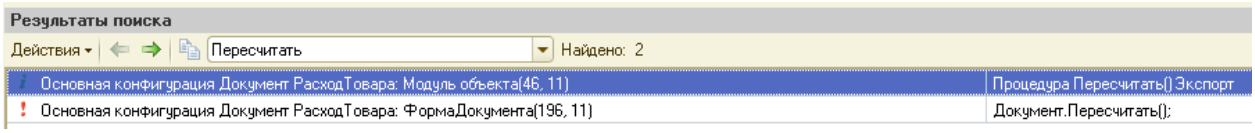
А в модуле формы этого документа – следующий обработчик:

```
&НаСервере
Процедура ПересчитатьНаСервере()

    Документ = РеквизитФормыВЗначение("Объект");
    Документ.Пересчитать();
    ЗначениеВРеквизитФормы(Документ, "Объект");

    //дальнейшая обработка...

КонецПроцедуры
```

Пиктограмма с красным восклицательным знаком в окне результатов поиска означает, что однозначно и точно определить использование в строке кода процедуры *Пересчитать()* системе не удалось.

Это связано с тем, что система не может автоматически определить тип переменной *Документ* после выполнения функции *РеквизитФормыВЗначение()*.

Механизм контекстной подсказки в этом случае также не предлагает возможные варианты при нажатии точки после переменной *Документ* либо при нажатии сочетания клавиш Ctrl+Пробел.



Переименование процедуры в модуле формы при помощи команды рефакторинга также приводит к замене ссылки на обработчик в свойствах элементов формы и командах.

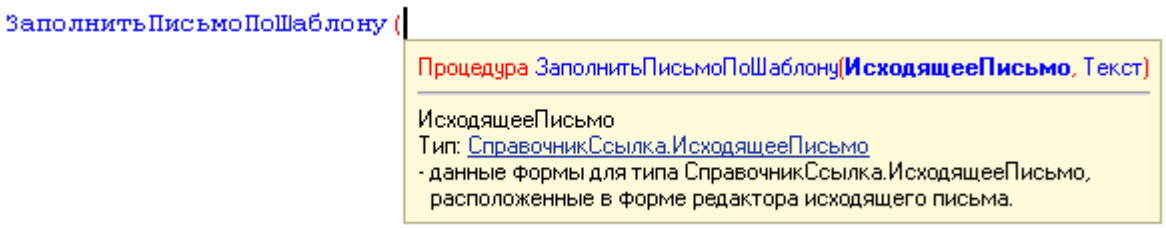
3. Создать описание функции

Команда создает перед процедурой или функцией комментарий, который будет корректно восприниматься механизмом контекстной подсказки.

```
// Процедура - Заполнить письмо по шаблону
//
// Параметры:
//  ИсходящееПисьмо      -       -
//  Текст             -       -
Процедура ЗаполнитьПисьмоПоШаблону(ИсходящееПисьмо, Текст) Экспорт
    //...
КонецПроцедуры
```

Система создает заготовку комментария, в который необходимо вставить типы параметров и пояснения.

Тогда можно будет при написании кода воспользоваться расширенной подсказкой.



4. Создать обработку оповещения

Эта команда становится доступной в контекстном меню, когда курсор установлен на имени метода, после которого следует открывающаяся скобка.

Причем это такие методы, как *ПоказатьВопрос()*, *ПоказатьПредупреждение()*, *ПоказатьВводЧисла()* и прочие блокирующие аналоги модальных методов.

Рассмотрим пример. Начнем писать клиентский обработчик команды, установим курсор на встречающийся метод *ПоказатьВопрос()*, вызовем команду “Создать обработчик оповещения”:

```
&НаКлиенте
Процедура ЗаполнитьМатериалы(Команда)

    ПоказатьВопрос(

КонецПроцедуры
```

В результате система сформирует следующий программный код:

```
&НаКлиенте
Процедура ЗаполнитьМатериалы(Команда)

    ПоказатьВопрос(Новый ОписаниеОповещения("ЗаполнитьМатериалыЗавершение",
ЭтотОбъект));

КонецПроцедуры

&НаКлиенте
Процедура ЗаполнитьМатериалыЗавершение(РезультатВопроса, ДополнительныеПараметры)
Экспорт


КонецПроцедуры
```

5. Преобразовать модальный вызов

Эта команда преобразует выделенный фрагмент кода, содержащий модальный метод, с использованием его асинхронного аналога. Рассмотрим несколько примеров.

Преобразуем вызов метода *Предупреждение()*:

```
&НаКлиенте
Процедура НовыйОбработчик()
    А = 1;
    Предупреждение("Текст");
    А = 2;
КонецПроцедуры // НовыйОбработчик()
```


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

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

После применения указанной команды программный код примет следующий вид:

```
&НаКлиенте
Процедура НовыйОбработчик()
    А = 1;
ПоказатьПредупреждение(Новый ОписаниеОповещения("НовыйОбработчикЗавершение",
ЭтотОбъект), "Текст");
КонецПроцедуры

&НаКлиенте
Процедура НовыйОбработчикЗавершение(ДополнительныеПараметры) Экспорт
    А = 2;
КонецПроцедуры
```

Усложним пример. Рассмотрим использование модальной функции и условного оператора:

```
&НаКлиенте
Процедура НовыйОбработчик()

    Ответ = Вопрос("Табличная часть будет очищена. Продолжить?",
РежимДиалогаВопрос.ДаНет);
    Если Ответ = КодВозвратаДиалога.Да Тогда
     //алгоритм заполнения
    КонецЕсли;

КонецПроцедуры
```

После преобразования модального вызова получаем:

```
&НаКлиенте
Процедура НовыйОбработчик()

    Ответ = Неопределено;

    ПоказатьВопрос(Новый ОписаниеОповещения("НовыйОбработчикЗавершение", ЭтотОбъект),
"Табличная часть будет очищена. Продолжить?", РежимДиалогаВопрос.ДаНет);

КонецПроцедуры

&НаКлиенте
Процедура НовыйОбработчикЗавершение(РезультатВопроса, ДополнительныеПараметры) Экспорт

    Ответ = РезультатВопроса;
    Если Ответ = КодВозвратаДиалога.Да Тогда
        //алгоритм заполнения
    КонецЕсли;

КонецПроцедуры
```

Следует подчеркнуть в получившемся фрагменте инициализацию переменной *Ответ*.

6. Преобразовать в асинхронную процедуру

В рассмотренных выше примерах преобразованию подвергались методы, имеющие свои асинхронные аналоги. Например, *Вопрос()* и *ПоказатьВопрос()*, *Предупреждение()* и *ПоказатьПредупреждение()*.

Однако если модальный вызов расположен внутри процедуры, которая в свою очередь располагается внутри еще одной процедуры, то в таком случае весь вызов процедуры с модальным методом внутри будет модальным.

А значит, его надо заменить на “асинхронный аналог”, только не тот, который существует во встроенном языке, а на наш собственный, разработанный метод.

Для этого и предназначена еще одна команда подменю “Рефакторинг” - “Преобразовать в асинхронную процедуру”. Поясним на примере процедуры, вызывающей другую процедуру с модальной функцией внутри:

```
&НаКлиенте
Процедура НовыйОбработчик()
    А = 1;
    ВложеннаяПроцедура();
    А = 2;
КонецПроцедуры

&НаКлиенте
Процедура ВложеннаяПроцедура()
    Предупреждение("Текст");
КонецПроцедуры
```

Устанавливаем курсор на объявление процедуры *ВложеннаяПроцедура()*, выполняем преобразование в асинхронную процедуру. Система строит нам следующий код:

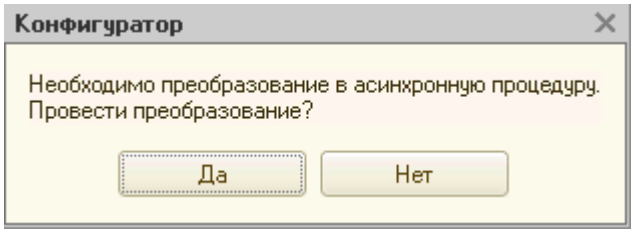
```
&НаКлиенте
Процедура НовыйОбработчик(Знач Оповещение)
    А = 1;
    ВложеннаяПроцедура(Новый ОписаниеОповещения("НовыйОбработчикЗавершение",
ЭтотОбъект, Новый Структура("Оповещение", Оповещение)));

КонецПроцедуры

&НаКлиенте
Процедура НовыйОбработчикЗавершение(Результат, ДополнительныеПараметры) Экспорт
    Оповещение = ДополнительныеПараметры.Оповещение;
    А = 2;
    ВыполнитьОбработкуОповещения(Оповещение);
КонецПроцедуры

&НаКлиенте
Процедура ВложеннаяПроцедура(Знач Оповещение)
    Предупреждение("Текст");
    ВыполнитьОбработкуОповещения(Оповещение);
КонецПроцедуры
```

Обратите внимание на добавленный системой метод *ВыполнитьОбработкуОповещения()*, который используется в реализации процедур, которые внутри себя могут открывать блокирующие окна, но при этом должны вернуть свой результат в вызывающие процедуры.

Следует помнить, что непосредственной задачей преобразования в асинхронную процедуру является преобразование последовательности вызовов выбранной процедуры к асинхронному виду, однако вызовы, расположенные в самой этой процедуре, не изменяются.

Именно поэтому метод *Предупреждение()* не подвергся замене. Это нужно сделать после преобразования в асинхронную процедуру, вызвав отдельно команду “Преобразовать модальный вызов”.

Если в исходном фрагменте кода на строке, содержащей *Предупреждение()*, выполнить команду “Преобразовать модальный вызов”, то система спросит:



Результат получится следующий:

```
&НаКлиенте
Процедура НовыйОбработчик(Знач Оповещение)
    А = 1;
    ВложеннаяПроцедура(Новый ОписаниеОповещения("НовыйОбработчикЗавершение",
ЭтотОбъект, Новый Структура("Оповещение", Оповещение)));

КонецПроцедуры

&НаКлиенте
Процедура НовыйОбработчикЗавершение(Результат, ДополнительныеПараметры) Экспорт

    Оповещение = ДополнительныеПараметры.Оповещение;
    А = 2;
    ВыполнитьОбработкуОповещения(Оповещение);

КонецПроцедуры

&НаКлиенте
Процедура ВложеннаяПроцедура(Знач Оповещение)
    ПоказатьПредупреждение(Новый ОписаниеОповещения("ВложеннаяПроцедураЗавершение",
ЭтотОбъект, Новый Структура("Оповещение", Оповещение)), "Текст");

КонецПроцедуры
```

```
&НаКлиенте
Процедура ВложеннаяПроцедураЗавершение(ДополнительныеПараметры) Экспорт

    Оповещение = ДополнительныеПараметры.Оповещение;
        ВыполнитьОбработкуОповещения(Оповещение);

КонецПроцедуры
```

7. Выделить в асинхронную процедуру

Эта команда преобразует выделенный участок кода в процедуру или функцию, преобразуя при этом выделяемый метод к асинхронному виду.

В отличие от предыдущего пункта данная команда является “составной”: сначала выделенный участок кода переносится в новую процедуру, имя которой пользователь вводит в диалоговом окне.

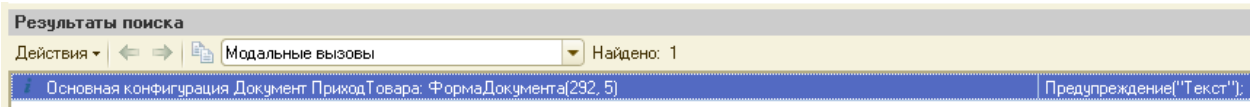
Затем выполняются действия, аналогичные тому, если бы пользователь щелкнул на заголовке только что созданной процедуры правой кнопкой мыши, а затем нажал “Преобразовать в асинхронную процедуру”.

8. Найти модальные вызовы модуля

Описанные выше команды работают с отдельным методом или выделенным участком кода.

Были реализованы процедуры, обрабатывающие модуль целиком, например, поиск модальных вызовов внутри всего модуля.

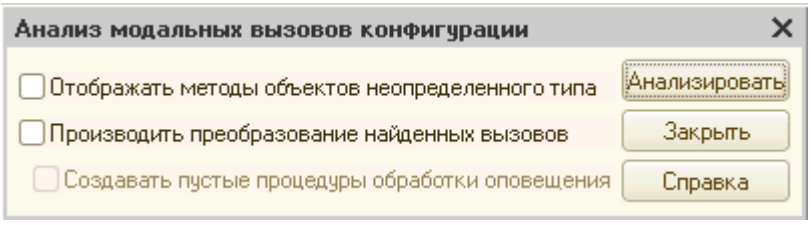
Найденные строки кода будут выведены в окно с результатами поиска:



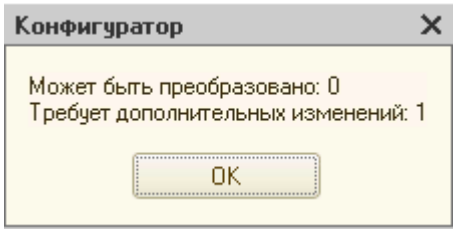
9. Преобразовать модальные вызовы модуля

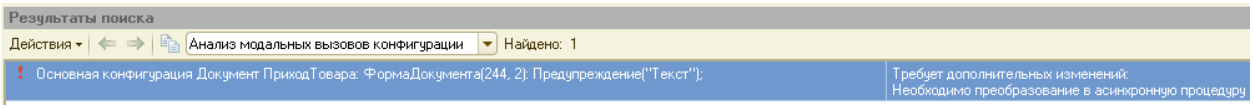
Эта команда выполняет преобразования в открытом модуле, но только тех вызовов, которые не требуют участия разработчика.

Также в главном меню расположена команда (Конфигурация - Рефакторинг - Анализировать модальные вызовы конфигурации).



Она также выполняет поиск модальных вызовов, только в рамках всей конфигурации целиком, проверяет, можно ли преобразовать модальные вызовы автоматически.





Ханевич Василий

г. Калининград

Дополнительные материалы

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