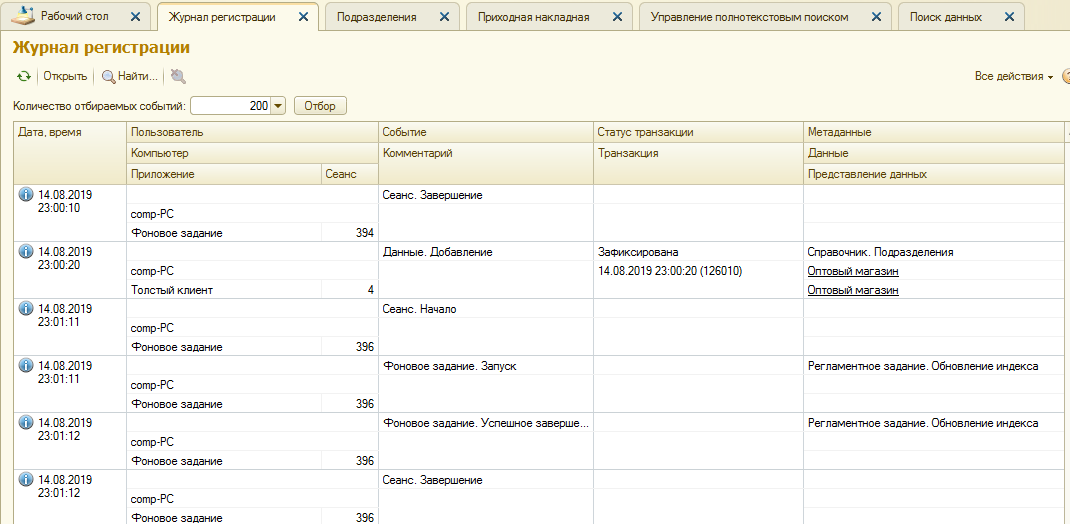
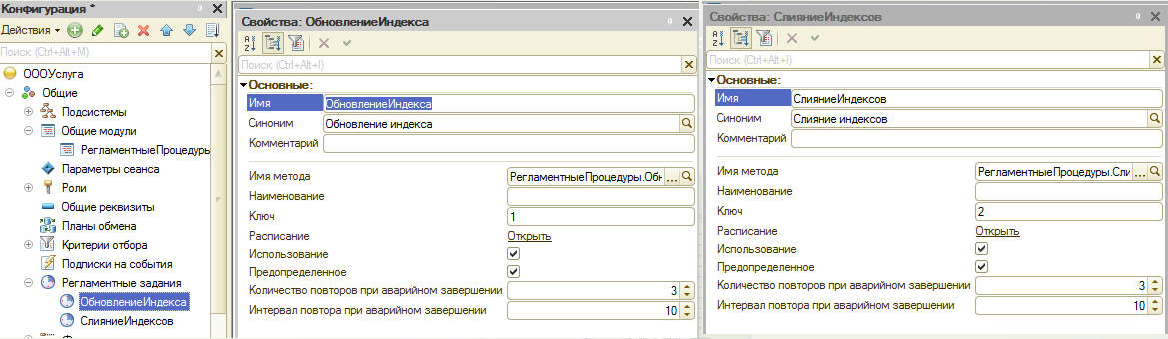
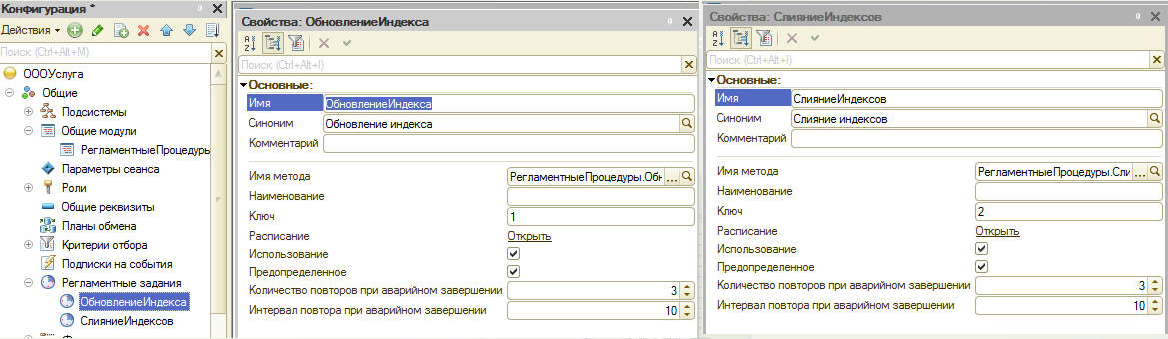
**AllowExecuteScheduledJobs**



**14**

**13**

COM (Component Object Model) – модель компонентных объектов Microsoft. Стандартный механизм, включающий интерфейсы, с помощью которых одни объекты предоставляют свои сервисы другим. Является основой многих объектных технологий, в том числе OLE и ActiveX. Другой перевод: многокомпонентная модель объектов.

DCOM (Distributed Component Object Model) – распределенная модель компонентных объектов. Расширение модели COM фирмы Microsoft, ориентированное на поддержку и интеграцию распределенных объектных приложений, функционирующих в сети.

COM представляет собой основанную на объектах клиент-серверную модель, разработанную Microsoft для обеспечения взаимодействия между компонентами программного обеспечения и приложениями. Microsoft расширила эту технологию добавлением элементов ActiveX, которые представляют результат развития технологий OLE и OCX. OCX (OLE Custom eXtension) – это программируемые компоненты-приложения с интерфейсом на базе OLE, позволяющим легко включать их в другие приложения. С 1996 года они стали называться ActiveX.

Сейчас Microsoft предлагает использовать термин Active technologies вместо ActiveX, включая в новые технологии такие составляющие:

· Active Documents (активные документы)

· ActiveX (управляющие элементы)

· Active Scripting controls

· Automation (автоматизация, прежде известная как OLE Automation)

Ключевым аспектом COM является то, что эта технология обеспечивает связь между клиентами и серверами посредством ***интерфейсов***. Именно интерфейс предоставляет клиенту способ "узнать" у сервера, какие именно возможности он поддерживает на этапе выполнения. Для расширения возможностей сервера необходимо просто добавить новый интерфейс к существующим.

Delphi предоставляет программисту мастеров (wizards) и классы, которые облегчают разработку приложений, основанных на технологиях COM. Мастера позволяют создавать:

· простые COM–совместимые классы для использования в одном приложении;

· полновесные серверы COM;

· серверы автоматизации (Automation servers) и контроллеры автоматизации (Automation controller);

· управляющие элементы ActiveX;

· активные формы ActiveForms.

**Технология COM как спецификация и реализация.** COM является одновременно и спецификацией и реализацией. Спецификация COM определяет правила создания объектов и их взаимодействия, способ связи между объектами. В соответствии со спецификацией объекты COM могут быть написаны на различных языках, выполняться в адресном пространстве различных процессов и на разнообразных платформах. До тех пор, пока объекты полностью соответствуют спецификации, они могут взаимодействовать. Это позволяет объединять унаследованный код как компонент с новыми компонентами, разработанными в любом из объектно-ориентированных языков.

COM как реализация представляет собой библиотеку (файлы OLE32.dll OLEAut32.dll), которая предоставляет ряд основных служб, поддерживающих описанные спецификации. Библиотека COM содержит набор стандартных интерфейсов, определяющих основную функциональность объектов COM, и небольшой набор функций API, разработанных для целей создания и управления объектами COM.

Интерфейсы объектов Delphi, как и язык Object Pascal, соответствуют COM спецификации. Реализация COM в Delphi называется DAX (Delphi ActiveX framework – базовая структура (каркас) элементов ActiveX Delphi). Основная часть реализации этой структуры находится в модуле AxCtrls.

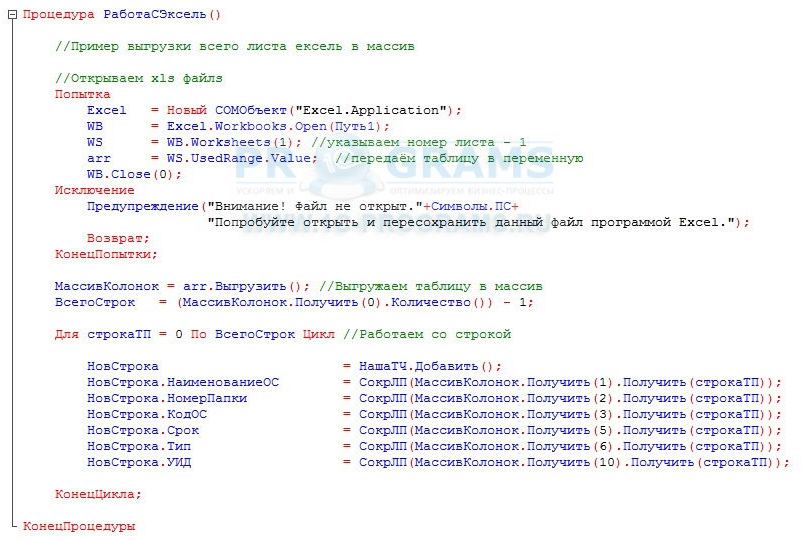
Когда программист использует мастеров Delphi или объекты библиотеки VCL в своем приложении, он использует реализацию COM спецификаций. Кроме того, Delphi предоставляет ряд упаковщиков (wrapper) для таких служб COM, которые не реализуются непосредственно, например, активные документы (Active Documents). Эти упаковщики включены в модуль ComObj

**Расширения COM (COM extensions).** Так как COM является развивающейся технологией, она может быть расширена за рамки базисных служб. COM является основой для других технологий, таких как автоматизация (Automation – вместо прежнего термина OLE Automation), элементы ActiveX и активные документы (Active Documents).

Кроме того, в настоящее время возможно создание таких объектов COM, которые могут взаимодействовать с Microsoft Transaction Server (MTS). MTS – это система обработки транзакций, предназначенная для построения, развертывания и управления большими Intranet и Internet приложениями-серверами. Хотя MTS архитектурно и не является частью COM, она разработана для расширения возможностей COM в большой, распределенной среде.

Delphi снабжает программиста мастерами, облегчающими разработку приложений, которые объединяют упомянутые выше технологии – Automation, ActiveX, Active Documents и MTS.

<https://makrus.ru/chastye-voprosy-po-1s/id-1s-zagruzka-iz-excel-i-vygruzka-iz-1s-v-excel.html>



|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Номер | Наименование | Артикул | Размер | Ед измерения | Страна |
| 1 | пальто | П-123 | 50 | шт | Россия |
| 2 | костюм | К-456 | 46 | шт | Китай |
| 3 | платье | Пл-23 | 42 | шт | Китай |
| 4 | блуза | Б-12 | 44 | шт | Корея |
| 5 | юбка | Ю-89 | 46 | шт | Россия |
| 6 | кардиган | Кр-76 | 52 | шт | Китай |

**Фрагмент Кода**

&НаСервере

Процедура ЗаполнитьНастройкиКолонокСправочника(ТЗ)

МетаданныеИсточника = ПолучитьМетаданныеИсточника();

Если МетаданныеИсточника = Неопределено Тогда

Возврат;

КонецЕсли;

Если МетаданныеИсточника.ДлинаКода > 0 Тогда

ЗагружаемыйРеквизит = ТЗ.Добавить();

ЗагружаемыйРеквизит.ИмяРеквизита = "Код";

ЗагружаемыйРеквизит.ПредставлениеРеквизита = "Код";

ЗагружаемыйРеквизит.МожетБытьПолемПоиска = Истина;

Если МетаданныеИсточника.ТипКода = Метаданные.СвойстваОбъектов.ТипКодаСправочника.Строка Тогда

ЗагружаемыйРеквизит.ОписаниеТипов = Новый ОписаниеТипов("Строка", , Новый КвалификаторыСтроки(МетаданныеИсточника.ДлинаКода));

Иначе

ЗагружаемыйРеквизит.ОписаниеТипов = Новый ОписаниеТипов("Число", , , Новый КвалификаторыЧисла(МетаданныеИсточника.ДлинаКода));

КонецЕсли;

Для каждого Реквизит Из МетаданныеИсточника.Реквизиты Цикл

Если Не Реквизит.Использование = Метаданные.СвойстваОбъектов.ИспользованиеРеквизита.ДляГруппы Тогда

ЗагружаемыйРеквизит = ТЗ.Добавить();

ЗагружаемыйРеквизит.ИмяРеквизита = Реквизит.Имя;

ЗагружаемыйРеквизит.ПредставлениеРеквизита = Реквизит.Представление();

ЗагружаемыйРеквизит.МожетБытьПолемПоиска = Не Реквизит.Индексирование = Метаданные.СвойстваОбъектов.Индексирование.НеИндексировать;

ЗагружаемыйРеквизит.ОписаниеТипов = МетаданныеИсточника.Реквизиты[ЗагружаемыйРеквизит.ИмяРеквизита].Тип;

КонецЕсли;

КонецЦикла;

Для каждого ЗагружаемыйРеквизит Из ТЗ Цикл

СписокВыбора = ПолучитьСписокИменПредставлений(ЗагружаемыйРеквизит.ОписаниеТипов);

ЗагружаемыйРеквизит.ИскатьПо = ?(СписокВыбора.Количество() = 0, "",СписокВыбора[0].Значение);

СписокВыбора = ПолучитьСписокСвязейПоВладельцу(ЗагружаемыйРеквизит.ОписаниеТипов, ТЗ);

ЗагружаемыйРеквизит.СвязьПоВладельцу = ?(СписокВыбора.Количество() = 0, "",СписокВыбора[0].Значение);

СписокВыбора = ПолучитьСписокСвязейПоТипу(ЗагружаемыйРеквизит, ТЗ);

Если СписокВыбора.Количество() = 0 Тогда

ЗагружаемыйРеквизит.СвязьПоТипу = "";

ЗагружаемыйРеквизит.ЭлементСвязиПоТипу = 0;

Иначе

ЗагружаемыйРеквизит.СвязьПоТипу = СписокВыбора[0].Значение;

Если Найти(ЗагружаемыйРеквизит.ИмяРеквизита,"3") <> 0 Тогда

ЗагружаемыйРеквизит.ЭлементСвязиПоТипу = 3;

ИначеЕсли Найти(ЗагружаемыйРеквизит.ИмяРеквизита,"2") <> 0 Тогда

ЗагружаемыйРеквизит.ЭлементСвязиПоТипу = 2;

Иначе

ЗагружаемыйРеквизит.ЭлементСвязиПоТипу = 1;

КонецЕсли;

КонецЕсли;

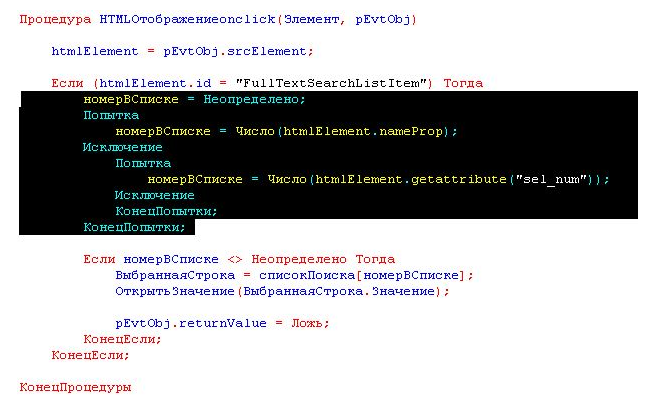
ЗагружаемыйРеквизит.ЗначениеПоУмолчанию = ЗагружаемыйРеквизит.ОписаниеТипов.ПривестиЗначение(Неопределено);

ЗагружаемыйРеквизит.ДоступныеТипы = ЗагружаемыйРеквизит.ОписаниеТипов;

ЗагружаемыйРеквизит.РежимЗагрузки = "Искать";

КонецЦикла;

КонецПроцедуры // ()



Фоновые и регламентные задания в файловой информационной базе Во многих типовых конфигурациях на платформе 8.2 существовала константа, в которой указывался пользователь, под которым выполнялись задания в файловом варианте. Для увеличения нажмите на изображение. При начале работы системы под этим пользователем происходило подключение обработчика ожидания, который с определенной периодичностью вызывал метод встроенного языка ВыполнитьОбработкуЗаданий(). Такой подход создавал дополнительные затруднения и неудобства для использования регламентных заданий в файловой базе. Это могло быть особенно критично при работе с такими конфигурациями, как “Комплексная автоматизация” или “Управление производственным предприятием”, в которых регламентные задания играют довольно важную роль.

Если ЭтоФайловаяИБ Тогда ПользовательДляВыполненияРеглЗаданий = Константы.ПользовательДляВыполненияРегламентныхЗаданийВФайловомВарианте.Получить(); Если глЗначениеПеременной(“глТекущийПользователь”) = ПользовательДляВыполненияРеглЗаданий Тогда // с интервалом секунд вызываем процедуру работы с регламентными заданиями ПоддержкаРегламентныхЗаданиеДляФайловойВерсии(); ИнтервалДляОпроса = Константы.ИнтервалДляОпросаРегламентныхЗаданийВФайловомВарианте.Получить(); Если ИнтервалДляОпроса = Неопределено ИЛИ ИнтервалДляОпроса = 0 Тогда ИнтервалДляОпроса = 60; КонецЕсли; ПодключитьОбработчикОжидания(“ПоддержкаРегламентныхЗаданиеДляФайловойВерсии”, ИнтервалДляОпроса); КонецЕсли; КонецЕсли;

В платформе 8.3 реализована работа фоновых и регламентных заданий в файловой информационной базе без использования вызова метода глобального контекста ВыполнитьОбработкуЗаданий(). Фоновые и регламентные задания выполняются непосредственно в одном из клиентских приложений (тонкий, толстый клиент) или расширением веб-сервера. Фоновое задание, запущенное каким-либо сеансом, выполняется тем же клиентским приложением, которое его инициировало. Фоновые задания выполняются последовательно, т.е. на одном клиентском приложении в один момент времени может выполняться только одно фоновое задание (в том числе и регламентное). Регламентные задания выполняются только одним клиентским приложением. Для управления запуском регламентных заданий предназначен ключ командной строки /AllowExecuteScheduledJobs.

Регламентные задания выполняются первым по порядку запуска клиентским приложением, которому не запрещено выполнение регламентных заданий (для запрещения сеанс должен быть запущен с параметром командной строки /AllowExecuteScheduledJobs -Off). Когда этот сеанс завершается, выполнение переходит к какому-либо из оставшихся запущенными сеансов. Если в строке запуска клиентского приложения явно указана необходимость запуска регламентных заданий (в командной строке указано /AllowExecuteScheduledJobs -Force), то регламентные задания начинают выполняться на нем, независимо от наличия других сеансов. Обработка регламентных заданий происходит один раз в 60 секунд. Заметим, что регламентные задания прикладного решения (в файловом варианте) начинали выполняться не ранее, чем через 1 минуту после запуска клиентского приложения. Начиная с редакции 8.3.8 это время увеличили вдвое, и поэтому регламентные задания начинают выполняться максимум через 2 минуты после запуска клиента (в файловом варианте). Такое решение обусловлено уменьшением задержки при старте приложения. Кроме того, в релизе 8.3.7 была добавлена возможность быстрого определения того, что текущий сеанс является сеансом фонового задания. Это реализовано за счет нового метода глобального контекста ПолучитьТекущийСеансИнформационнойБазы(), а также нового метода у сеанса информационной базы ПолучитьФоновоеЗадание(). Ниже показан код, который иллюстрирует вышеописанное:.

Теперь разберем подробнее создание задания. Нам необходимо заполнить форму следующим образом: **Имя, синоним и комментарий.** Учтите, что правила и принципы заполнения этих полей аналогичны для всех объектов конфигурации и описывать их еще раз нет необходимости.

**Имя метода**. Процедура общего модуля, которая максимально точно описывает необходимый алгоритм. **Ключ.** Позволяет сгруппировать несколько разных регламентных заданий 1С. Одновременно может быть запущено только одно задание с одинаковым значением ключа, но само значение может быть произвольным.

**Расписание**. Один из важнейших элементов, т.к. именно по данному расписанию будет выполняться задание.

**Использование**. Определяет активность задания.

**Предопределенное**. Предопределенное – определяет, что при запуске 1С такое задание будет создано в единственном экземпляре с указанным в конфигураторе расписанием. Не предопределенные задания появляются в момент наступления расписания программно.

**Количество повторов.** Если в процессе выполнения возникла исключительная ситуация, происходит перезапуск, но чтобы не возникла бесконечная зацикленность, программа ограничивает перезапуски по количеству.

**Интервал повтора**. Время в секундах между повторными запусками аварийно завершенных заданий.

Источник: <https://1c.programs93.ru/articles/reglamentnye-fonovye-zadanija-v-1s/> Поддержка, обновление, внедрение и доработка 1С - 1С аутсорсинг в Москве и МО

