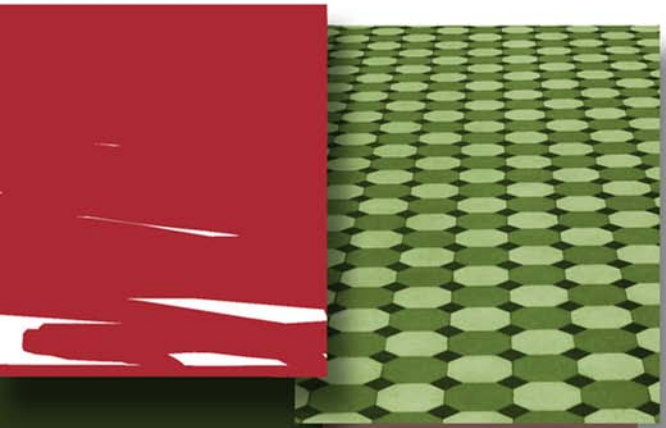
ВСЕВОЛОД НЕСВИЖСКИЙ

bhv®

1С: ПРЕДПРИЯТИЕ 8.0

ПРИЕМЫ ПРОГРАММИРОВАНИЯ



ПОДРОБНОЕ ОПИСАНИЕ РАБОТЫ С БАЗОВЫМИ ОБЪЕКТАМИ

АНАЛИЗ И ОБРАБОТКА ДАННЫХ

ПОСТРОЕНИЕ ОТЧЕТОВ И ПЕЧАТНЫХ ФОРМ

ВЗАИМОДЕЙСТВИЕ С ВНЕШНИМИ БАЗАМИ ДАННЫХ SQL

ПРАВИЛА ПОСТРОЕНИЯ ЗАПРОСОВ

PRO

ПРОФЕССИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

+CD

Всеволод Несвижский

1С: ПРЕДПРИЯТИЕ 8.0

ПРИЕМЫ ПРОГРАММИРОВАНИЯ

Санкт-Петербург

«БХВ-Петербург»

2007

УДК 681.3.06
ББК 32.973.26-018.2
Н55

Несвижский В.

Н55 1С:Предприятие 8.0. Приемы программирования. — СПб.: БХВ-Петербург, 2007. — 512 с.: ил. + CD-ROM — (Профессиональное программирование)

ISBN 978-5-9775-0089-0

Книга полностью построена на реальных примерах и задачах, решаемых 1С-программистами в повседневной работе. Представленные приемы программирования универсальны и применимы в любых существующих конфигурациях системы 1С:Предприятие 8.0. Рассмотрены наиболее важные и часто используемые объекты конфигурации: документы, справочники, регистры накопления, регистры сведений, отчеты, макеты и др. Особое внимание уделено разработке печатных документов, применению построителя отчетов и анализу данных. Подробно описано, как подключить существующие внешние базы данных SQL к системе 1С:Предприятие 8.0. Отличительной особенностью книги является большое количество примеров с подробными комментариями. Исходные тексты всех примеров содержатся на прилагаемом компакт-диске.

Для 1С-программистов

УДК 681.3.06
ББК 32.973.26-018.2

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зам. главного редактора	*Евгений Рыбаков*
Зав. редакцией	*Григорий Добин*
Редактор	*Елена Кашлакова*
Компьютерная верстка	*Натальи Караваевой*
Корректор	*Виктория Пиотровская*
Дизайн серии	*Инны Тачиной*
Оформление обложки	*Елены Беляевой*
Зав. производством	*Николай Тверских*

Лицензия ИД № 02429 от 24.07.00. Подписано в печать 22.06.07.
Формат $70\times100^{1}/_{16}$. Печать офсетная. Усл. печ. л. 41,28.
Тираж 2500 экз. Заказ №
"БХВ-Петербург", 194354, Санкт-Петербург, ул. Есенина, 5Б.

Санитарно-эпидемиологическое заключение на продукцию
№ 77.99.02.953.Д.006421.11.04 от 11.11.2004 г. выдано Федеральной службой
по надзору в сфере защиты прав потребителей и благополучия человека.

Отпечатано с готовых диапозитивов
в ГУП "Типография "Наука"
199034, Санкт-Петербург, 9 линия, 12

ISBN 978-5-9775-0089-0

Оглавление

Введение

Не так давно фирма 1С порадовала многих своих поклонников, выпустив обновленную версию своей, не побоюсь сказать, самой популярной в СНГ программы для управления предприятием и ведения бухгалтерского учета. Новая система "1С:Предприятие 8.0" не просто заменила полюбившуюся "1С:Предприятие 7.7", а намного превзошла ее по своим возможностям. Разработчиками была проделана поистине титаническая работа, в результате которой появилась очень мощная, настраиваемая, гибкая и многофункциональная система, позволяющая в короткие сроки с нуля создавать готовые решения для автоматизации различных видов хозяйственной деятельности или использовать уже имеющиеся. Во-первых, было предложено несколько вариантов работы системы: однопользовательский, файловый и клиент-серверный. Последний наиболее полно отвечает современным требованиям эффективной работы большого количества пользователей в распределенной среде. Во-вторых, система может решать широкий круг вопросов, начиная от автоматизации торговой и производственной деятельности, бухгалтерского учета и заканчивая планированием, бюджетированием и анализом финансовой деятельности. В-третьих, быстрота и удивительная гибкость разработки новых законченных решений, имеющих все необходимые средства для сбора, хранения и анализа данных. Все это послужило основной причиной растущей популярности системы "1С:Предприятие 8.0", а также создания данной книги.

Для кого написана эта книга? Конечно для разработчиков, уже использующих новую систему, а также для тех, кто собирается переходить на новый продукт. Книга является сугубо практическим руководством по программированию, построению отчетов и печатных форм, а также взаимодействию с другими базами данных, написанных на MSSQL (языке запросов фирмы Microsoft). Не секрет, что встроенная справка и предлагаемая разработчиками документация особо не изобилуют примерами, что иногда приводит в тупик начинающих программистов. В данной книге автор надеется восполнить

этот существенный пробел и помочь читателям научиться быстро и эффективно применять богатые возможности встроенного языка для работы с общими, а главное — прикладными объектами, наиболее востребованными в условиях современного рынка.

Как уже было сказано, система "1С:Предприятие 8.0" имеет мощный встроенный язык, внешним видом немного напоминающий добрый старый C++. К основным особенностям нового языка можно отнести:

- код полностью объектно-ориентированный;
- как минимум, возможность выбора русского или английского языка для написания кода;
- наличие синтакс-помощника, различных конструкторов и шаблонов;
- проверка синтаксиса;
- отсутствие явной типизации переменных;
- удобное автоматическое форматирование процедур, функций, условий и циклов;
- контекстная подсказка.

Для применения всего перечисленного фирма 1С предусмотрела специальное средство разработки — конфигуратор, который решает не только все эти задачи, но и позволяет создавать или корректировать интерфейс программы, выполнять административные функции и управлять распределением доступа к программе. Кроме того, конфигуратор служит единственной точкой входа для создания и поддержки конфигурации. Конфигурация представляет собой иерархическую структуру, состоящую из общих и прикладных объектов. К общим объектам относятся:

- *подсистемы* — позволяют группировать различные объекты конфигурации;
- *общие модули* — предназначены для хранения глобальных процедур, функций и переменных, доступных из любого объекта конфигурации;
- *роли* — позволяют определить права доступа пользователей к различным объектам конфигурации;
- *интерфейсы* — помогают установить и распределить взаимодействие пользователей с программой посредством панелей инструментов, обычных и контекстных меню;
- *критерии отбора* — предназначены для организации отборов данных по заданным критериям;
- *общие формы* — представляют собой стандартные формы интерфейса, неоднократно используемые различными объектами конфигурации для решения однотипных повторяющихся задач;

- *общие макеты* — предназначены для использования в качестве стандартных печатных форм, которые могут вызываться из различных объектов конфигурации;
- *общие картинки* — содержат графические файлы для организации красивых и понятных интерфейсов пользователя, например, вместо текстовых надписей на панели инструментов;
- *стили* — позволяют создавать предопределенные варианты форматирования и оформления для различных элементов управления;
- *языки* — данные объекты помогают организовать многоязыковую поддержку конфигурации.

К прикладным объектам можно отнести:

- *константы* — позволяют хранить и использовать в различных объектах конфигурации определенные постоянные значения;
- *справочники* — позволяют определить общую условно-постоянную информацию для обращения из других объектов конфигурации;
- *документы* — содержат всевозможные виды документов, необходимых для решения поставленных задач;
- *журналы документов* — позволяют отбирать и группировать различные типы документов по определенным признакам;
- *перечисления* — предназначены для хранения наборов значений одного типа и подстановки в программный код;
- *отчеты* — позволяют создавать удобные для последующего анализа представления данных;
- *обработки* — удобные объекты, которые могут применяться для решения всевозможных специфических или сервисных задач, не реализованных в имеющихся модулях конфигурации;
- *планы видов характеристик* — позволяют назначать дополнительные характеристики и свойства для товаров, а также организовать аналитический учет в бухгалтерии;
- *планы счетов* — предназначены для хранения счетов управленческого и бухгалтерского учета;
- *планы видов расчета* — предназначены для организации общих правил расчета, например при пересчете налогов;
- *регистры сведений* — позволяют хранить какую-либо заданную информацию о различных объектах конфигурации, в том числе и с учетом времени;

- *регистры накопления* — предназначены для накопления предопределенной информации, выраженной числовыми значениями и с учетом времени;
- регистры бухгалтерии — позволяют хранить данные проводок по определенным планам счетов.

Существуют и другие объекты, о которых можно получить информацию в справке 1С. Они не имеют принципиального значения в контексте данной книги, поэтому здесь не рассматриваются.

На базе перечисленных общих и прикладных объектов строится конфигурация, отвечающая различным задачам управления предприятием и ведением бухгалтерского учета. Существуют и готовые решения, позволяющие в короткие строки организовать полноценный товарооборот и учет торговых операций. К ним можно отнести: УПП — управление производственным предприятием, УТ — управление торговлей, УП — управление персоналом, УС — управление складом и другие. УПП представляет собой наиболее полное решение на базе "1С:Предприятие 8.0" и включает в себя следующие подсистемы: управление отношениями с покупателями и поставщиками, управление оборудованием и ремонтами, управление розничной торговлей, расчет зарплаты, планирование продаж и закупок, управление производством, управление денежными средствами, бюджетирование, бухгалтерский и налоговый учет, управление заказами, управление складом, управление взаиморасчетами, управление персоналом. УПП позволяет в короткие сроки организовать полноценную программную систему управления и анализа любым торговым или производственным предприятием. Однако в реальной жизни может потребоваться гораздо более скромный набор возможностей, по сравнению с имеющимся в УПП. Для этих целей имеет смысл применить более "легкую" конфигурацию. Например, для организации складского учета вполне достаточным будет установить только систему управления складом. Этим можно сэкономить финансовые затраты и упростить работу пользователей. Поэтому, выбирая вариант конфигурации, следует заранее определить решаемый круг задач, которые будут выполняться на данном конкретном предприятии или участке. Выбранный вариант конфигурации будет включать в себя только те объекты, которые необходимы, но разработчик имеет возможность добавлять свои собственные в рамках общей структуры, описанной ранее, а также дорабатывать существующие объекты. При этом возникает круг вопросов, связанных с поддержкой и обновлением конфигурации, не рассматриваемый в данной книге. Более подробную информацию об этом можно получить в службе поддержки фирмы 1С.

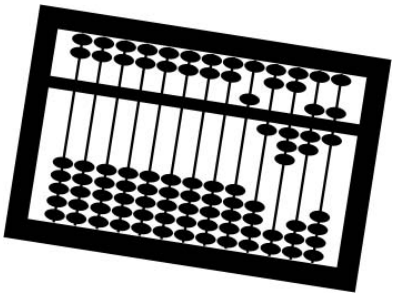
Исходя из всего сказанного, можно определить основные темы данного руководства:

1. Приемы программирования документов.
2. Приемы программирования регистров накопления.
3. Приемы программирования регистров сведений.
4. Анализ данных.
5. Отчеты.
6. Объекты конфигурации.
7. Печатные документы.
8. Взаимодействие 1С и баз данных SQL.

Основой упор в книге делается на практическое программирование свойств и методов различных объектов, написание и оптимизацию запросов. Подразумевается, что читатель знаком с языком 1С, имеет общее представление о конфигураторе и базовых объектах, умеет пользоваться синтакс-помощником и встроенной справкой. В качестве языка программирования выбран русский. Сделано это не случайно, а по причине более удобного восприятия и комфортного написания текстов исходных кодов. Все примеры, рассмотренные в книге, имеются на прилагаемом компакт-диске.

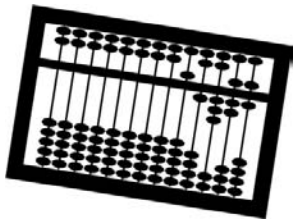
Программные требования

Для полноценной работы с книгой, необходимо иметь на компьютере установленную лицензионную программу "1С:Предприятие 8.0" версии не ниже 8.0.3.16. Кроме этого, рекомендуется иметь одну из следующих конфигураций: управление производственным предприятием, управление производством или управление торговлей. Материалы, рассматриваемые в книге, целиком содержатся только в первой из перечисленных конфигураций, поэтому не стоит удивляться, не обнаружив какие-либо объекты у себя. Для работы с примерами, демонстрирующими подключение к внешним базам данных, следует установить Microsoft SQL Server 2000 или выше.



Часть I

Приемы программирования документов



Глава 1

Создание, запись, проведение, блокировка, удаление

Документы являются неотъемлемой частью любой конфигурации. Они составляют основу прикладной направленности продукта, его возможности и удобство работы пользователей с предложенным интерфейсом. Грамотно организованный документ позволяет упростить и ускорить процесс формирования данных в программе, а также помогает провести в последующем более полный анализ результатов работы, влияющий в целом на результаты работы всего предприятия.

Готовые конфигурации уже содержат в себе необходимый набор документов, отвечающий поставленным задачам по торгово-производственным операциям и бухгалтерскому учету. В этой части приводятся приемы программирования различных документов, эффективное использование их свойств, методов и событий. Поскольку в разных конфигурациях могут присутствовать как одинаковые, так и абсолютно непохожие типы документов, что обусловлено решением разного круга задач, читателям необходимо учитывать данный момент. Здесь описываются только наиболее интересные и часто используемые документы, которые могут быть в одной конфигурации (например, Управление производственным предприятием) и отсутствовать в другой, направленной на какую-то одну область учета (например, Управление персоналом).

Читатели познакомятся с программированием документов посредством встроенной объектной модели 1С и языка запросов. Первый вариант наиболее приемлем для создания и обновления документов в базе, их проведения и удаления, получения и установки свойств, обработки событий. Второй позволяет получать массивы однотипных документов для их последующей обработки, задействовав при этом меньшие ресурсы системы, что очень актуально для больших баз данных. Хорошим правилом программирования считается применение, где это возможно, языка запросов.

Существующая в "1С:Предприятие 8.0" объектная модель предоставляет программисту базовые объекты для управления документами:

- ДокументыМенеджер — предоставляет доступ ко всем имеющимся в конфигурации типам документов;
- ДокументОбъект — позволяет получить доступ непосредственно к документу для его изменения;
- ДокументМенеджер — предназначен для получения доступа к определенному виду документа, его формам и макетам;
- ДокументСсылка — данный объект позволяет получить ссылку на документ без возможности изменения последнего;
- ДокументСписок — объект необходим для управления списком однотипных документов, отображаемых в табличном виде;
- ДокументВыборка — позволяет организовать выборку массива документов из базы данных для их последующей обработки.

Каждый из этих объектов имеет свои преимущества и недостатки. Например, изменить существующий в базе документ можно через объект ДокументОбъект, а получить выборку — через ДокументВыборка. Для использования ссылки на документ (без модификации самого объекта документа) предпочтительней выбрать ДокументСсылка. ДокументСписок помогает управлять настройками отборов и упорядочивания отображаемого списка документов, с которыми работает пользователь.

Любой документ сможет содержать в себе реквизиты, табличные части, формы и макеты. Дата создания документа и номер являются предопределенными реквизитами и назначаются системой автоматически. Структуру документа можно еще представить следующим образом:

- *Шапка* — представляет, как правило, собой общие сведения о документе, такие как номер, дата создания, наименование организации и контрагента, номер и дата договора, валюта документа, наименование склада, сумму документа и НДС (налог на добавленную стоимость);
- *Список товаров* — табличная часть, которая содержит сведения о товарах, отражаемых в документе. К таким сведениям, в первую очередь, относятся наименование товара, характеристика, количество, единица измерения, цена за единицу товара, сумма НДС, стоимость;
- *Список услуг* — дополнительная табличная часть, описывающая различные услуги, предоставляемые дополнительно к товарной части или самостоятельно. Наиболее распространенным видом услуг являются транспортные расходы на перевозку грузов. К сведениям об услугах относятся: наименование, количество, единица измерения, количество, стоимость;

- *Дополнительные сведения* — различные добавочные сведения, в зависимости от вида документа, а также подразделение и имя ответственного за создание документа.

А теперь на практике рассмотрим приемы программирования документов, используя перечисленные объекты.

Создание нового документа

Первый документ, который мы сформируем, будет ЗаказПокупателя. Этот вид документа является отправной точкой в торговых операциях и служит для формирования заявки покупателя (иначе говоря, контрагента) на приобретение у нас каких-либо товаров или предоставления ему определенных услуг.

Для создания нового документа достаточно использовать метод объекта ДокументМенеджер `СоздатьДокумент`, как показано в листинге 1.1.

Листинг 1.1. Создание нового документа

```
НовыйЗаказПокупателя = Документы.ЗаказПокупателя.СоздатьДокумент();
// Установим текущую дату создания документа
НовыйЗаказПокупателя.Дата = ТекущаяДата();
// Установим свой номер для документа
НовыйЗаказПокупателя.Номер = "Документ 1";
// Определим вид операции
НовыйЗаказПокупателя.ВидОперации =
                              Перечисления.ВидыОперацийЗаказПокупателя;
// Добавим комментарий
НовыйЗаказПокупателя.Комментарий = "Образец создания нового документа";
```

В приведенном примере создается новый документ ЗаказПокупателя, где дате создания присваивается значение текущей даты (встроенная функция `ТекущаяДата`). Кроме того, задается свой номер документа, комментарий и устанавливается вид операции. Вид операции представляет собой предопределенное в конфигурации значение перечисления `ВидыОперацийЗаказПокупателя`. Номер документа может быть числовым или строковым типом. Это зависит от установленных в конфигураторе свойств документа. Строковый тип гораздо предпочтительней, поскольку позволяет вводить буквы и цифры одновременно, что очень важно для предприятий, использующих собственную систему нумерации документов.

Как видите, создать новый документ очень просто, но может понадобиться не только создать документ, но и вывести на экран форму документа, например, для самостоятельного заполнения ее пользователем. Решение данной задачи на примере документа ВнутреннийЗаказ показано в листинге 1.2.

Листинг 1.2. Получение и вывод на экран основной формы документа

```
НовыйВнутреннийЗаказ = Документы.ВнутреннийЗаказ.СоздатьДокумент();
// Установим номер документа, используя встроенную функцию
НовыйВнутреннийЗаказ.УстановитьНовыйНомер();
// Получим основную форму документа, определенную в конфигураторе
ФормаВнутреннийЗаказ =
                НовыйВнутреннийЗаказ.ПолучитьФорму("ФормаДокумента");
// Выводим форму документа на экран
ФормаВнутреннийЗаказ.Открыть();

// Последние две строки кода можно заменить одной
ФормаВнутреннийЗаказ =
НовыйВнутреннийЗаказ.ПолучитьФорму("ФормаДокумента").Открыть();
```

После выполнения кода листинга будет создан и появится на экране монитора новый документ ВнутреннийЗаказ. Пользователь сможет заполнить все необходимые данные и сохранить документ в базе. Для выбора номера мы применили доступный для данного объекта метод `УстановитьНовыйНомер`. Получение формы реализуется через специальный метод объекта `Получить-Форму`. В качестве аргумента этого метода следует указать имя основной формы, назначенного этому объекту документа в конфигурации. Кроме того, метод `ПолучитьФорму` имеет еще два необязательных параметра: ссылка на родительскую форму и уникальный ключ для последующего быстрого обращения к форме.

Как правило, у документа есть только одна основная форма, но могут быть еще несколько дополнительных, например, форма отображения списка данного вида документов, форма выбора и другие. При создании новой конфигурации разработчик сам задает имена основной и дополнительных форм, а в типовых конфигурациях они уже заданы, поэтому необходимо изучить структуру документа перед его использованием. Более подробно работа с формами будет рассмотрена в *главе 4*.

Запись и проведение документа

После того как создан новый документ, его следует сохранить в базе данных, а также, если необходимо, провести. Процесс сохранения и проведения документа реализован в одном методе — `Записать`. Этот метод имеет два необязательных аргумента. Первый позволяет установить режим записи (запись, проведение или отмена проведения), а второй определяет режим проведения (оперативный или не оперативный). Приведем пример сохранения нового документа в листинге 1.3.

Листинг 1.3. Запись документа

```
НовыйАвансовыйОтчет = Документы.АвансовыйОтчет.СоздатьДокумент();
// Установим текущую дату создания документа
НовыйАвансовыйОтчет.Дата = ТекущаяДата();
// Установим свой номер для документа
НовыйАвансовыйОтчет.Номер = "00004";
// Запишем документ
НовыйАвансовыйОтчет.Записать();
```

Итак, мы создали новый документ АвансовыйОтчет, установили дату и номер документа, а затем сохранили его в базе данных, используя метод `Записать`. После того как документ записан, может потребоваться его провести. Термин "проведение" подразумевает выполнение движений документа в системе по связанным с ним регистрам. Как правило, каждый документ хранит в себе различные сведения, которые можно использовать впоследствии для построения отчетов и анализа данных. Чтобы упростить задачу, фирма 1С разработала так называемые *регистры*, которые отслеживают и накапливают информацию об изменениях в документах. Подробнее о назначении и работе с регистрами будет рассказано в *части II книги*. Сейчас достаточно понять, что при проведении документа предопределенные сведения о нем записываются в связанные регистры, заданные на этапе создания конфигурации. Пример проведения документа представлен в листинге 1.4.

Листинг 1.4. Запись и проведение документа

```
НовыйЗаказПоставщику = Документы.ЗаказПоставщику.СоздатьДокумент();
// Установим текущую дату создания документа
НовыйЗаказПоставщику.Дата = НачалоДня(ТекущаяДата());
// Запишем и проведем документ
```

```
НовыйЗаказПоставщику.Записать(РежимЗаписиДокумента.Проведение);
// Отменим проведение документа
НовыйЗаказПоставщику.Записать(РежимЗаписиДокумента.ОтменаПроведения);
```

Сначала мы создали новый документ, затем провели и записали его в базу данных, а после этого отменили проведение. Проведение документа может быть выполнено оперативно (оперативный режим подразумевает проведение документа текущей датой и временем) или "задним числом". Пример оперативного проведения документа представлен в листинге 1.5.

Листинг 1.5. Оперативное проведение документа

```
НовыйРеализацияТоваров = Документы.РеализацияТоваров.СоздатьДокумент();
// Установим текущую дату создания документа
НовыйРеализацияТоваров.Дата = ТекущаяДата();
// Выберем из справочника валюту
НовыйРеализацияТоваров.ВалютаДокумента =
                        Справочники.Валюты.НайтиПоНаименованию("RUR");
// Запишем и проведем документ оперативно
НовыйРеализацияТоваров.Записать(РежимЗаписиДокумента.Проведение,
                                РежимПроведенияДокумента.Оперативный);
```

Если документ уже существует в базе данных, он будет просто перепроведен. При создании нового — документ вначале записывается в базу, а потом проводится.

Блокировка документа

Как правило, время от времени в многопользовательской системе возникает необходимость внести какие-либо изменения в документ. Поскольку один и тот же документ могут открыть одновременно несколько пользователей, следует перед внесением изменений попытаться заблокировать его, чтобы гарантировать сохранение вносимой информации. Заметьте, речь идет о попытке блокировки, поскольку объект уже может быть заблокирован другим пользователем. Хорошим правилом будет выполнение предварительной проверки состояния блокировки с последующим принятием решения — блокировать документ или подождать, пока он освободится. Рассмотрим пример блокировки документа в листинге 1.6.

Листинг 1.6. Управление блокировкой документа

```
// Создадим новый документ
НовыйПоступлениеТоваров = Документы.ПоступлениеТоваров.СоздатьДокумент();
// Заполним реквизиты документа
НовыйПоступлениеТоваров.Номер = "001";
// . . .
// Сохраним его в базе данных
НовыйПоступлениеТоваров.Записать();
// Теперь получим ссылку на созданный нами документ
ПоступлениеТоваровСсылка =
                    Документы.ПоступлениеТоваров.НайтиПоНомеру("001");
// Получаем объект документа по ссылке
ПоступлениеТоваровОбъект = ПоступлениеТоваровСсылка.ПолучитьОбъект();
// Блокируем документ
Попытка
        ПоступлениеТоваровОбъект.Заблокировать();
        // Вносим изменения в документ, например, меняем номер
        ПоступлениеТоваровОбъект.Номер = "002";
        // Записываем данные в базу
        ПоступлениеТоваровОбъект.Записать();
        // Разблокируем документ
        ПоступлениеТоваровОбъект.Разблокировать();
Исключение
        // Выводим сообщение об ошибке
        Предупреждение ("Не удалось заблокировать документ!");
КонецПопытки;
```

Итак, вначале мы создали новый документ, заполнили данными и сохранили в базе данных. После этого получили ссылку на существующий документ, воспользовавшись методом `НайтиПоНомеру`, где в качестве аргумента передали номер искомого документа. Получив ссылку, мы вызвали метод `ПолучитьОбъект`, чтобы внести изменения в документ. Конечно, можно было использовать созданный вначале объект, а не идти столь длинным путем, но это было сделано умышленно и только в качестве демонстрации работы с блокировкой. Далее мы создали обработчик ошибок, где и выполнили попытку блокировки. В случае неудачи пользователь получил бы сообщение об ошибке. В случае успешной блокировки мы изменили номер документа, записали обновленный документ в базу и разблокировали его, вызвав метод `Разблокировать`.

Следует заметить, что для проверки наличия блокировки имеется специальный метод `Заблокирован`, который возвращает истинное или ложное значение, в зависимости от состояния блокировки. Однако, как показывает практика, его применение не всегда дает правильный результат, поэтому рекомендуется использовать в таких ситуациях обработчик ошибок.

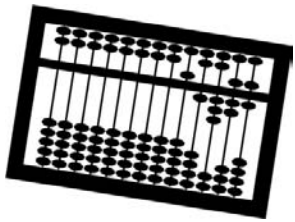
Удаление документа

В "1С:Предприятие 8.0" удаление документа разделено на две составляющие: пометка объекта на удаление и непосредственное удаление. В первом случае документ не удаляется из базы, а только устанавливается признак того, что он больше не нужен. При этом с проведенного документа снимаются все проводки (движения по регистрам), но документ остается в базе и отображается в общем списке. В любой момент его можно снова восстановить и провести. Во втором случае документ безвозвратно удаляется из базы без возможности восстановления. При этом все имеющиеся связи документа с другими объектами конфигурации сохраняются, что может привести в дальнейшем к многочисленным ошибкам. Рассмотрим пример удаления документа в листинге 1.7.

Листинг 1.7. Удаление документа

```
// Создадим новый документ
НовыйСписаниеТоваров = Документы.СписаниеТоваров.СоздатьДокумент();
// Заполним дату документа
НовыйСписаниеТоваров.Дата = ТекущаяДата();
// Запишем документ в базу
НовыйСписаниеТоваров.Записать();
// Пометим документ к удалению
Если НЕ НовыйСписаниеТоваров.ПометкаУдаления Тогда
       НовыйСписаниеТоваров.УстановитьПометкуУдаления;
КонецЕсли;
// или удалим документ непосредственно
НовыйСписаниеТоваров.Удалить();
```

Как видно из примера, был создан новый документ и сохранен в базе. После этого мы вызвали метод `УстановитьПометкуУдаления`, предварительно проверив наличие признака удаления (свойство `ПометкаУдаления`). После этого документ все еще находится в базе. Чтобы удалить его окончательно, нужно применить метод `Удалить`.



Глава 2

Выбор и поиск

Неотъемлемой частью любой базы данных является возможность выборки и поиска данных. Поскольку документы интенсивно накапливаются в результате работы пользователей, размер базы увеличивается очень быстро. Для эффективной работы требуются средства, позволяющие быстро и правильно находить нужные записи, при этом как можно меньше загружать имеющиеся процессорные мощности. Любая современная система управления базами данных в полной мере отвечает этим требованиям. Не стала исключением и 1С, предоставив разработчикам и конечным пользователям широкие возможности по выбору и поиску в базе отдельных записей. Однако следует помнить, что без грамотного подхода конечный результат может оказаться не таким оптимистичным, как предполагается. Связано это, прежде всего, с определенными требованиями, налагаемыми на использование методов поиска и выборки данных. К ним относятся грамотное применение критериев (условий) поиска. Именно с помощью критериев поиска можно сократить время получения данных, их конечную значимость и информативность.

В этой главе мы рассмотрим, как можно программно организовать выборку одного или множества документов, найти строго определенный документ в базе данных и как правильно использовать различные условия поиска.

Как уже было сказано в *главе 1*, система предоставляет разработчикам готовые базовые объекты ДокументВыборка и ДокументМенеджер, с помощью которых можно легко организовать выборку большого количества документов для последующей обработки и модификации. Для начала попробуем выбрать документы, удовлетворяющие определенным условиям, как показано в листинге 2.1.

Листинг 2.1. Выборка документов в заданном диапазоне дат

```
// Задаем критерии выборки по дате
НачальнаяДата = НачалоДня(ТекущаяДата());
КонечнаяДата  = КонецДня(ТекущаяДата());
// Делаем выборку документов из базы за текущий день
ВыборкаЗаказПокупателя =
Документы.ЗаказПокупателя.Выбрать(НачальнаяДата, КонечнаяДата);
// Организуем цикл для обработки найденных документов
Пока ВыборкаЗаказПокупателя.Следующий() Цикл
        // Выводим в окно сообщений номер найденного документа
        Сообщить("Номер документа: " + ВыборкаЗаказПокупателя.Номер);
        // Выводим в окно сообщений признак проведения
        Сообщить("Проведен: " + ВыборкаЗаказПокупателя.Проведен);
        // Выводим в окно сообщений признак пометки на удаление
        Сообщить("Помечен на удаление: " + ВыборкаЗаказПокупателя.Номер);
        // Получаем объект документа
        ОбъектЗаказПокупателя = ВыборкаЗаказПокупателя.ПолучитьОбъект();
        // Теперь мы можем корректировать документ
        Если ОбъектЗаказПокупателя.Проведен Тогда
                // Если документ проведен, отменим проведение
                ОбъектЗаказПокупателя.Записать(
                                РежимЗаписиДокумента.ОтменаПроведения);
        КонецЕсли;
КонецЦикла;
```

Итак, мы задали условие выбора документов — диапазон дат в пределах текущего дня. После этого с помощью метода `Выбрать` делаем выборку документов из базы данных. Первый параметр задает начальную дату поиска, а второй — конечную дату. Далее организуем цикл для обхода всех найденных записей, предварительно позиционируя курсор специальным методом `Следующий` на первую позицию. Теперь нам доступны для чтения номер и дата документа, а также мы можем узнать, проведен ли документ или помечен на удаление. Кроме этого, можно прочитать значения всех реквизитов и получить доступ к табличным частям документа, если они у него есть. Как вы заметили, все перечисленные данные о документе мы можем только читать. Чтобы иметь возможность внести изменения, нам пришлось воспользоваться методом `ПолучитьОбъект`, после чего документ стал доступен не только для чтения, но и для записи. Таким образом, сделав несложный цикл, можно кор-

ректировать набор документов одного вида, в данном примере ЗаказПокупателя. Кстати, начальную и конечную даты можно и не задавать, поскольку это необязательные аргументы. В этом случае мы получим все имеющиеся в базе документы указанного вида. Сразу замечу, что это не лучшее решение, поскольку использование объектов конфигурации для выборки данных и без того существенно замедляет работу всей системы в целом. Чем точнее будут указанные условия выбора, тем быстрее будет обработан ваш запрос.

Дополнительно в методе `Выбрать` можно настроить отбор по одному определенному полю, для которого в конфигурации установлено свойство индексирования. Это может быть реквизит документа или табличной части. Условие отбора задается при помощи третьего аргумента. И, наконец, имеется возможность осуществить упорядочивание по какому-либо полю, для которого также в конфигурации установлено свойство индексирования. Значение упорядочивания задается в четвертом аргументе. Рассмотрим пример, использующий отбор и упорядочивание, приведенный в листинге 2.2.

Листинг 2.2. Выборка документов с отбором и упорядочиванием

```
// Задаем условие отбора по виду операции
УсловиеОтбора = Новый Структура("ВидОперации",
                Перечисления.ВидыОперацийАвансовыйОтчет.АвансовыйОтчет);
// Задаем упорядочивание по дате создания документа
ПолеУпорядочивания = "Дата Убыв";
// Делаем выборку документов из базы
ВыборкаАвансовыйОтчет =
Документы.АвансовыйОтчет.Выбрать(, , УсловиеОтбора, ПолеУпорядочивания);
// Организуем цикл для обработки найденных документов
Пока ВыборкаАвансовыйОтчет.Следующий() Цикл
        // Выводим в окно сообщений номер найденного документа
        Сообщить("Номер документа: " + ВыборкаАвансовыйОтчет.Номер);
        // Выводим в окно сообщений признак проведения
        Сообщить("Проведен: " + ВыборкаАвансовыйОтчет.Проведен);
КонецЦикла;
```

Как видно из примера, в качестве условия отбора мы выбрали один из видов операций, заданных в конфигурации для данного типа документа, и передали его в виде структуры. Также мы определили способ упорядочивания записей — по дате создания документа в порядке убывания, т. е. начиная с самой последней даты.

Как уже говорилось, с помощью метода `Выбрать` можно получить не только реквизиты документа, но и реквизиты табличных частей. На примере документа РасходныйОрдерНаТовары попробуем прочитать значения реквизитов табличной части Товары, заданной для данного документа в конфигурации (листинг 2.3).

Листинг 2.3. Получение реквизитов табличной части документа

```
// Задаем упорядочивание по дате создания документа
ПолеУпорядочивания = "Дата Возр";
// Делаем выборку документов из базы
ВыборкаРасходныйОрдерНаТовары =
Документы.РасходныйОрдерНаТовары.Выбрать(, , , ПолеУпорядочивания);
// Организуем цикл для обработки найденных документов
Пока ВыборкаРасходныйОрдерНаТовары.Следующий() Цикл
    // Выводим в окно сообщений номер найденного документа
    Сообщить("Номер документа: " + ВыборкаРасходныйОрдерНаТовары.Номер);
    // Выводим в окно сообщений дату документа
    Сообщить("Дата создания: " + ВыборкаРасходныйОрдерНаТовары.Дата);
    // Получаем табличную часть Товары
    ТабЧастьТовары = ВыборкаРасходныйОрдерНаТовары.Товары;
    // Выводим в окно сообщений наименование номенклатуры
    Сообщить("Номенклатура: " + ТабЧастьТовары.Номенклатура);
    // Выводим в окно сообщений характеристику номенклатуры
    Сообщить("Характеристика номенклатуры: " +
                ТабЧастьТовары.ХарактеристикаНоменклатуры);
    // Выводим в окно сообщений количество товара
    Сообщить("Количество: " + ТабЧастьТовары.Количество);
    // Выводим в окно сообщений цену товара
    Сообщить("Цена: " + ТабЧастьТовары.Цена);
КонецЦикла;
```

Как видно из примера, мы сделали выборку всех имеющихся в базе документов РасходныйОрдерНаТовары с упорядочиванием по дате создания в порядке возрастания. После этого получили ссылку на табличную часть документа Товары и вывели в служебное окно сообщений несколько наиболее информативных реквизитов табличной части документа. Как видите, все оказалось достаточно просто и понятно. Замечу, что данные из табличной части можно не только читать, но и корректировать. Для этого достаточно получить объект

документа (см. листинг 2.1) и далее ссылку на данные табличной части. Думаю, вам не составит труда сделать это самостоятельно.

И последний вопрос, который мы разберем, касается поиска документов в базе. Имеется два основных метода для организации поиска какого-либо документа в базе: `НайтиПоНомеру` и `НайтиПоРеквизиту`. Первый из них позволяет найти документ по номеру и дате, а второй — по одному из реквизитов документа, заданных в конфигураторе. В листинге 1.5 уже приводился пример, использующий метод `НайтиПоНомеру`, где в качестве единственного аргумента передавался номер документа. Если документ существует, мы получим ссылку на него, иначе система возвратит пустую ссылку. Есть еще второй вариант этого метода, в котором задается не только номер документа, но и дата. Указанная дата будет связана с определенным диапазоном дат, в зависимости от установленного в конфигураторе признака контроля уникальности номеров и периодичности повторяемости одинаковых номеров документов. Периодичность задается в пределах одного дня, месяца, квартала или года. Другими словами, при указании, например, даты 20 мая 2007 г. и периода в 1 месяц, поиск искомого документа будет выполнен в диапазоне с 1 по 31 мая 2007 г. включительно. Рассмотрим пример, демонстрирующий сказанное (листинг 2.4).

Листинг 2.4. Поиск документа по номеру и дате

```
// Пытаемся найти документ и получить на него ссылку
СсылкаПеремещениеТоваров =
Документы.ПеремещениеТоваров.НайтиПоНомеру("00345", ТекущаяДата());
Если СсылкаПеремещениеТоваров.Пустая() Тогда
        // Искомый документ не найден в базе
        Вопрос("Документ с указанным номером не найден!"
                                    РежимДиалогаВопрос.OK, , , "Поиск");
Иначе
        // Используем ссылку на документ для создания копии
        НовыйПеремещениеТоваров = СсылкаПеремещениеТоваров.Скопировать();
        // Корректируем документ при необходимости
        НовыйПеремещениеТоваров.Номер = "00346";
        // . . .
        // Сохраняем новый документ в базе
        НовыйПеремещениеТоваров.Записать();
КонецЕсли;
```

В приведенном примере мы выполнили поиск документа по номеру и текущей дате. Если в конфигураторе свойство периодичности для данного вида документа установлено как месяц, система будет искать документ с указанным номером в диапазоне дат, начиная с начала месяца и заканчивая последним днем. Номер месяца берется из текущей даты. Если документ не найден, мы получим пустую ссылку и выведем на экран соответствующее сообщение. В случае существования искомого документа с помощью метода `Скопировать` создаем копию документа, с которой можно работать как с объектом, т. е. корректировать, записывать, удалять, проводить.

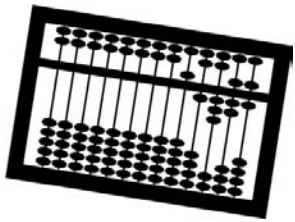
Метод `НайтиПоРеквизиту` позволяет найти документ, указав какой-либо реквизит, заданный в конфигураторе. В качестве первого аргумента указывается наименование самого реквизита, а во второй записывается его значение. Посмотрите, как это реализуется на примере листинга 2.5.

Листинг 2.5. Поиск документа по реквизиту

```
// Пытаемся найти документ и получить на него ссылку
СсылкаСписаниеТоваров =
Документы.СписаниеТоваров.НайтиПоРеквизиту("Ответственный",
                                            "Радкевич О.А.");
Если СсылкаСписаниеТоваров.Пустая() Тогда
        // Искомый документ не найден в базе
        Вопрос("Документ не найден!", РежимДиалогаВопрос.ОК, , , "Поиск");
Иначе
        // Используем ссылку на документ для получения объекта
        ОбъектСписаниеТоваров = СсылкаСписаниеТоваров.ПолучитьОбъект();
        // Выбираем нового ответственного
        ОбъектСписаниеТоваров.Ответственный =
                        Справочники.ФизическиеЛица.НайтиПоКоду("7");
        // Сохраняем документ в базе
        ОбъектСписаниеТоваров.Записать();
КонецЕсли;
```

Как видно из примера, вначале мы выполнили поиск документа, задав в качестве реквизита ответственного за создание документа. После этого, получив объект документа, изменили имя сотрудника и сохранили документ в базе.

В завершение хотелось бы подчеркнуть, что наибольшая эффективность выбора множества документов из базы достигается не с помощью объектов конфигурации, а посредством запросов, о которых будет подробно рассказано в *главе 8*. Описанные здесь методы выборки и поиска рекомендуется применять лишь для корректировки одиночных документов.



Глава 3

Получение ссылок

В предыдущих главах мы немного коснулись такого понятия, как ссылка на документ. Основным предназначением ссылок является получение (чтение) реквизитов документа и табличных частей, заданных на этапе проектирования в конфигураторе. Для тех читателей, которые знакомы с другими языками программирования (например, C++ или VB), смысл ссылок, думаю, вполне понятен. Для тех же, кто первый опыт программирования приобретает в системе 1С, поясню, что применение ссылки на какой-либо объект позволяет гораздо эффективнее использовать память и в целом существенно повысить скорость работы программы. Представьте себе, для чего требуется больше памяти: для получения всего объекта или для единственного адреса (ссылки) в памяти, который ссылается на этот объект. Конечно же, адрес, условно представляющий собою числовое значение, будет более правильным решением. Поскольку в 1С полноценно реализованы принципы объектно-ориентированного программирования, работа через ссылки является базовым звеном правильного подхода к программированию различных объектов конфигурации. Удобство ссылок трудно переоценить. Достаточно открыть любой справочник или документ в конфигураторе, чтобы убедиться, что многие реквизиты представляют собой так называемые ссылочные типы данных (например, реквизит `ВалютаДокумента` является ссылкой на справочник валют).

Получить ссылку на документ можно посредством следующих базовых объектов: ДокументВыборка, ДокументМенеджер, ДокументОбъект. Во всех случаях в результате мы получаем объект ДокументСсылка. Такое разнообразие открывает широкие возможности для получения информации о документе в любой удобный для разработчика момент и в различных ситуациях. Попытаемся подробно на примерах рассмотреть все эти варианты использования ссылок.