

Boxoft Image To PDF Demo. Purchase from www.Boxoft.com

to remove the watermark

Разработка приложений в Lotus Domino 7

2-е издание, исправленное

Грапенджизер Ф.

Сантана М.

Монсон Ф.

Чейз Л.

Ларссон Д.

Национальный Открытый Университет "ИНТУИТ"

2016

Разработка приложений в Lotus Domino 7

2-е издание, исправленное

Грапенджизер Ф.

Сантана М.

Монсон Ф.

Чейз Л.

Ларссон Д.

Национальный Открытый Университет "ИНТУИТ"

2016

Разработка приложений в Lotus Domino 7/ Ф. Грапенджизер, М. Сантана, Ф. Монсон, Л. Чейз, Д. Ларссон - М.: Национальный Открытый Университет "ИНТУИТ", 2016

Курс предназначен для того, чтобы показать разработчикам приложений, как использовать новейшие функции и возможности, появившиеся в IBM Lotus Notes, а также в IBM Lotus Domino 7.

В курсе рассмотрены функции разработки приложений, такие как интеграция с IBM DB2 Universal Database и Web-сервисы. Преимущества всех нововведений показаны на примере, в котором улучшается приложение для вымышленной компании.

(с) ООО "ИНТУИТ.РУ", 2008-2016

(с) Грапенджизер Ф., Сантана М., Монсон Ф., Чейз Л., Ларссон Д., 2008-2016

Разработка приложений в среде Lotus Domino 7

Лекция является вводной. В ней рассматриваются общие принципы разработки приложений в Lotus Domino 7

Разработка приложений в Lotus Domino 7 – существенный шаг вперед в функциональности, который позволяет пользователю создавать более мощные, легко распространяемые и масштабируемые приложения IBM Lotus Domino. С выходом этой версии IBM Lotus Domino Designer предоставляет разработчикам приложений больше объема для хранения данных и расширенные возможности работы с Universal Database (UDB) IBM DB2, интегрированной с Lotus Domino, преимущества улучшенного доступа к данным и работы с ними при помощи поддержки Web-сервисов, дополнительные элементы проектирования приложений, а также усовершенствованные инструменты диагностики и отладки приложений. В Lotus Notes и Domino 7 появились инструменты, позволяющие разработчикам создавать приложения, которые будут отвечать конкретным бизнес-целям, и в то же время, эти инструменты позволяют представлять приложения Lotus Domino потребителям в профессиональном и интуитивно понятном интерфейсе при просмотре любым способом, через Lotus Notes, Lotus Domino for Web Access или через интерфейс Web-браузера. К тому же внедренная в Lotus Notes и Domino 7 поддержка Web-сервисов позволяет таким приложениям, как Microsoft .NET, JavaTM, Java 2 Platform, Enterprise Edition (J2EE), а также многим другим взаимодействовать с Lotus Domino без дополнительного программирования.

В данном руководстве мы решили продемонстрировать новые возможности и элементы разработки приложений в среде Lotus Domino 7; данную задачу мы решили осуществить с помощью программ, приведенных в качестве примера, которые мы улучшали для вымышленной корпорации ITSO Electronics (будем считать, что команда, написавшая эту книгу, – консультанты этой компании). Мы демонстрируем разработчикам приложений, как разрабатывать в среде Lotus Domino приложение, которое использовало бы возможности взаимодействия и хранения базы данных DB2, интегрированной с Lotus Domino, а затем как увеличить возможность компании совершать сделки с внешними поставщиками и партнерами, интегрируя Web-сервисы в приложения. Все это необходимо будет выполнять с помощью новых

элементов разработки приложений, появившихся в Lotus Domino 7. Наконец, мы привели в этом курсе инструкции по использованию новых возможностей отладки приложений LotusScript и Java, а также возможности профилирования для отладки, диагностирования проблем и разрешения спорных моментов.

1.1 Сценарий ITSO Electronics

Как мы уже говорили, в следующих лекциях будут более подробно рассматриваться новейшие возможности и функции разработки приложений, появившиеся в Lotus Domino 7. Для примера в курсе будут приведены приложения, чтобы показать, как использовать каждую новую функцию, появившуюся в Lotus Domino 7 применительно к деятельности компании ITSO Electronics. Эти приложения можно загрузить с Web-сайта IBM Redbooks. Для получения более подробной информации см. приложение А, "Дополнительный материал".

Замечание Эти приложения предназначены только лишь для того, чтобы дать представление о новых возможностях и функциях разработки приложений, появившихся в Lotus Domino 7. Они не предназначены для использования в реальных приложениях.

Обзор приложения ITSO Electronics

Вымышленная компания ITSO Electronics использует Lotus Domino для поддержки работы отдела продаж. Через несколько лет они внедрили приложение, отслеживающее продажи для того, чтобы помочь сотрудникам отдела продаж взаимодействовать со своими покупателями. Доступ к базе данных можно получить с помощью клиента Lotus Notes или с помощью Web-браузера.

Существуют две основные базы данных в приложении, отслеживающем продажи ITSO Electronics: база данных *Sales* (Продажи) и база данных *Customers* (Покупатели). Рис 1.1 и 1.2 иллюстрируют пользовательский интерфейс, который применяется при доступе из Lotus Notes. Третья база данных, *Products* (Продукты), используется для хранения ключевых слов продуктов. Напрямую она не доступна для

персонала. Наконец, четвертая база данных была добавлена к существующей среде приложения для элементов Web-сервисов.

Замечание Все ссылки на базы данных с примерами, содержащими новые функции и описанными в данной книге, для удобства изучения вложены в приложение А, "Дополнительный материал".

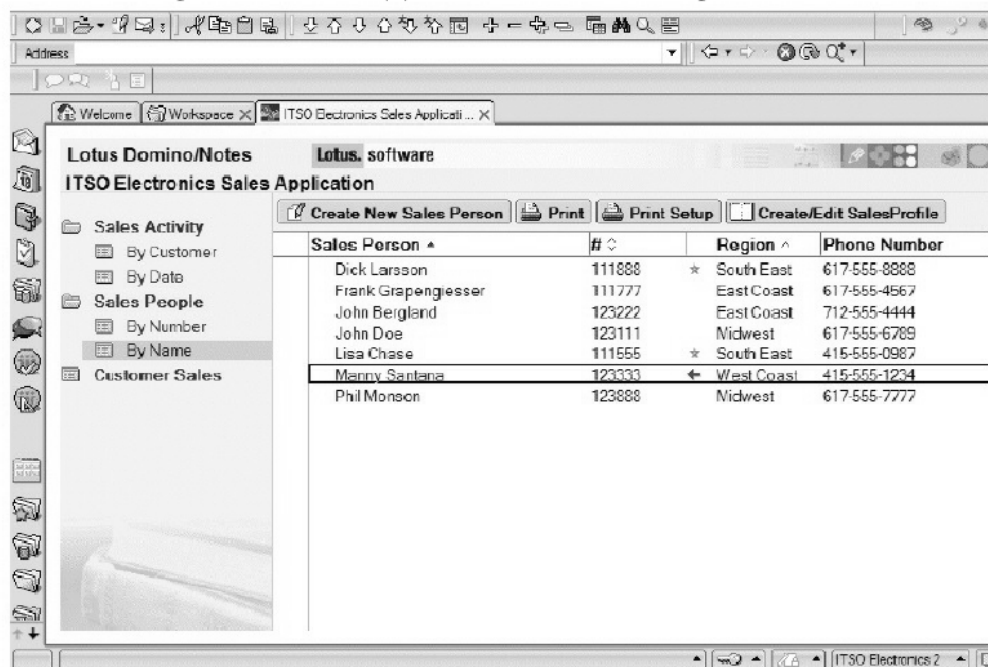


Рис. 1.1. Пользовательский интерфейс Lotus Notes для приложения, отслеживающего продажи, база данных Sales

Замечание Для применения Web-сервисов не требуется использование отдельной базы данных Lotus Domino, потому что Web-сервисы могут быть добавлены или объединены с любой базой данных Domino. Для наглядности мы использовали в данной книге отдельную базу данных Web Services.

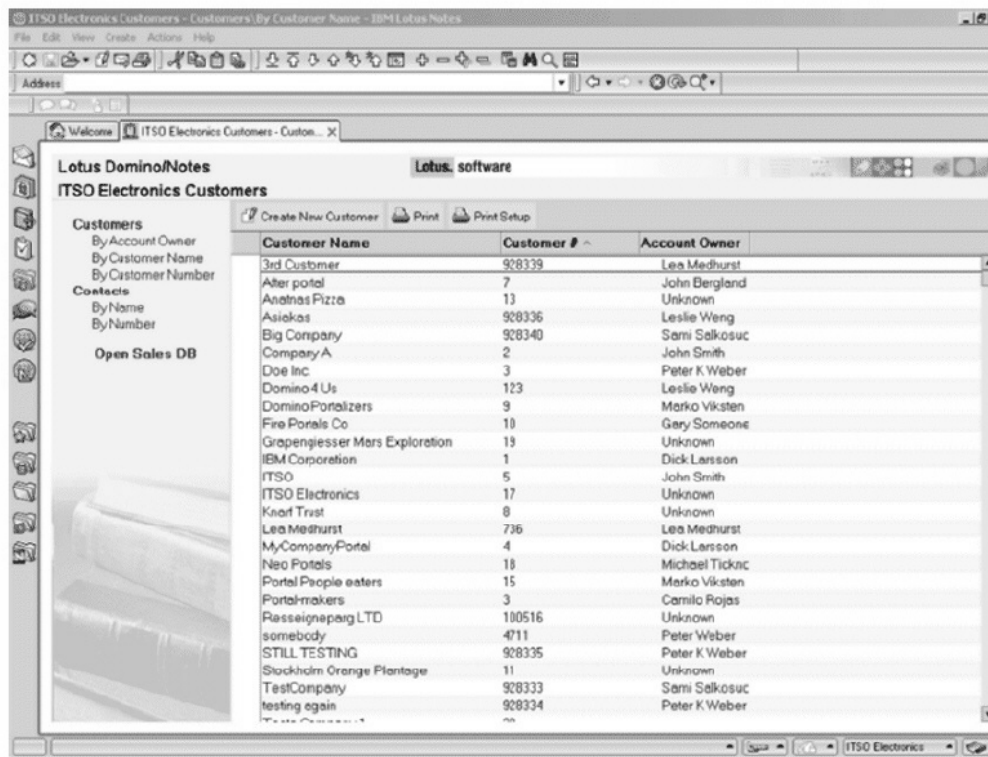


Рис. 1.2. Пользовательский интерфейс Lotus Notes для приложения, отслеживающего продажи, база данных Customers

Рис 1.3 иллюстрирует архитектуру приложения ITSO Electronics, в которой базы Sales, Customers, Products и Web Services располагаются на сервере DB2 UDB, но доступ к ним осуществляется через сервер Lotus Domino.

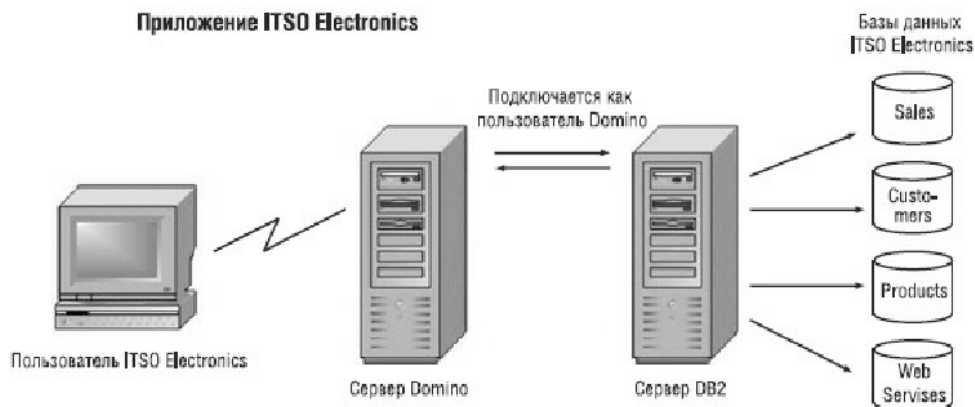


Рис. 1.3. Архитектура приложения ITSO Electronics

1.2 Как организована эта книга

Мы представляем вам разработку приложения ITSO Electronics в Lotus Domino 7 в четырех главах, написанных по следующему принципу.

В лекции 2, "Использование интеграции с DB2 для улучшения приложения ITSO Electronics", мы будем изучать расширение интеграции DB2 с Lotus Domino и то, как возможности хранения данных в DB2 и ее реляционные возможности смогут сделать приложение ITSO Electronics, написанное в среде Lotus Domino, значительно более мощным и функциональным. И, еще более важно, вы поймете, как самим сделать более функциональное приложение. Вопреки исторически сложившимся возможностям интеграции реляционных баз данных с Lotus Domino, используя Lotus Enterprise Integrator [®], мы исследуем достоинства архитектуры DB2 в разработке приложений Lotus Domino. Мы обсудим, как подготовиться к интеграции DB2 и Lotus Domino, DB2 Access Views и Query Views, DB2 federation, а также коснемся вопросов безопасности интеграции хранения данных DB2 с Lotus Domino. Также в эту главу включены советы по диагностике при установке и конфигурировании среды DB2.

Web-сервисы можно считать, пожалуй, наиболее интересным дополнением, появившимся в инструментарии разработки приложений Lotus Domino, и в лекции 3, "Улучшение работы приложения ITSO

Electronics с помощью Web-сервисов", мы рассмотрим технологию Web-сервисов и то, как она позволяет увеличить возможности приложения ITSO Electronics, сделанного в среде Lotus Domino. В то время как COM-интеграция предложила открытый доступ к Lotus Domino начиная с версий 5.0x, Web-сервисы перешагнули зависимость от модели доступа к данным, открыв разработчикам приложений Lotus Domino сервисную интеграцию. В этой лекции мы исследуем сервис-ориентированную архитектуру (SOA) и то, как Web-сервисы соответствуют плану SOA, а также уделим внимание языку описания Web-сервисов (WSDL), простому протоколу доступа к объектам (SOAP), процессу добавления элементов Web-сервисов к приложениям Lotus Domino, обращению с исключениями и ошибками, безопасности и употреблению Web-сервисов.

В Lotus Domino 7 появилось несколько новых мощных элементов разработки приложений, которые мы будем обсуждать и демонстрировать в лекции 4, "Применение новых элементов разработки в приложении ITSO Electronics", в которой мы будем улучшать приложение ITSO Electronics. Мы детально изучим добавления к языку формул, включая классы и объекты LotusScript и Java, а также действия, осуществляемые по щелчку правой кнопки мыши и возможности общих столбцов.

В лекции 5, "Диагностирование и отладка приложения ITSO Electronics", речь пойдет о новейших инструментах диагностирования и отладки приложений в среде Lotus Domino 7. Используя приложение ITSO Electronics в качестве примера, мы изучим методы оптимизации кода вашего приложения Lotus Notes, дополнительные возможности отладки LotusScript, а также новые возможности отладки Java, включая удаленную отладку.

1.3 Разработка приложений в Lotus Domino Designer

IBM Lotus Domino Designer представляет собой мощную среду разработки для создания бизнес-приложений, которые используют сервер Lotus Domino. Мы предполагаем, что если вы заинтересовались данным курсом, то, скорее всего, являетесь опытным разработчиком приложений в среде Lotus Domino и знакомы с особенностями разработки приложений и элементами разработки приложений Lotus

Domino Designer в Lotus Domino версии 6. Демонстрацию разработки приложения ITSO Electronics мы начнем с добавления DB2 как хранилища данных и реляционной среды при условии, что вы хорошо себе представляете, как построить приложение Lotus Domino без компонента DB2. Если вы не знакомы с разработкой приложений в среде Lotus Domino Designer, обратитесь к книге с техническими статьями Domino Designer 6: A Developer's Handbook, SG24-6854, и ознакомьтесь с элементами разработки приложений и функциями в Lotus Domino Designer. Руководство доступно по адресу: ссылка: <http://www.redbooks.ibm.com/abstracts/sg246854.html>.

Использование интеграции с DB2 для улучшения приложения ITSO Electronics

В этой лекции рассматриваются вопросы интеграции IBM Lotus Notes и Domino 7 с IBM DB2 Universal Database (UDB) на примере приложения ITSO Electronics

В этой лекции мы обсудим новые возможности, возникающие при интеграции IBM Lotus Notes и Domino 7 с IBM DB2 Universal Database (UDB). Там, где это необходимо, мы используем в качестве примера приложение ITSO Electronics для объяснения новых возможностей разработки приложений.

Ниже вы найдете информацию, в которой будут раскрыты следующие темы:

- Введение в интеграцию Domino и DB2.
- DB2 Access Views.
- Query Views.
- Работа с интегрированными данными DB2.
- Анализ безопасности DB2 и Lotus Domino.
- Поиск и устранение неисправностей в DB2 и Lotus Domino.

Замечание IBM Lotus Notes и Domino 7 с поддержкой DB2 были выпущены с ограниченной доступностью (Limited Availability); как таковые они не доступны для общего пользования, а рассчитаны только для оценки их возможностей. Для получения дополнительной информации о программе Limited Availability, а также другой важной информации, касающейся этой темы, посетите страницу в Интернете ссылка: <http://www.ibm.com/lotus/nsfdb2>.

2.1 Введение в интеграцию Domino и DB2

Lotus Notes и Domino имеют в арсенале широкий спектр средств для интеграции с реляционными базами данных, бизнес-системами, а также системами планирования и управления ресурсами предприятия (enterprise resource planning – ERP), такими как Lotus Enterprise Integrator, Domino Enterprise Connection Services, Data Connection Resources и Lotus

Connectors LotusScript Extensions. Lotus продолжает эти традиции с Notes и Domino 7, предоставляя в распоряжение разработчика новые децентрализованные возможности разработки приложений для интеграции с реляционными базами данных IBM DB2, которые одинаково могут применять как бизнес-пользователи, так и ведущие разработчики приложений. Лаборатории Lotus и DB2 в пределах корпорации IBM работали вместе для улучшения интеграционных возможностей, а сервер Domino 7 обеспечивает возможность выборочно определить новые или существующие базы данных Domino, так как в них включена поддержка DB2. Пользователи получают открытый доступ к базам данных Domino с поддержкой DB2 через сервер Domino 7, но сами базы данных хранятся в DB2. Таким образом, становится возможным в базе данных Domino с поддержкой DB2 использовать реляционные конструкции, а также данные из базы данных Domino с поддержкой DB2 в реляционных конструкциях. В то же время сохраняется поддержка функций Notes API, включая репликацию данных, маршрутизацию почты и административный доступ. Так как IBM DB2 является реляционной системой базы данных, данные с отдельных баз данных DB2, таблиц, видов и интегрированных объектов могут быть легко скомбинированы в новые наборы данных, используя общие поля данных и отношения. Все это позволяет разработчику работать с данными и представлять их другими, более удобными способами, и облегчает создание более интегрированных и значимых приложений в среде Domino и других средах, использующих данные Domino, данные DB2 и реляционные возможности. Более подробно эти возможности мы обсудим далее в этой главе.

До того как использовать новые интеграционные возможности Domino и DB2, вы должны установить сервер DB2 и сконфигурировать сервер Domino 7 для подключения к серверу DB2. В Lotus Domino 7 новые интеграционные возможности доступны в операционных системах Microsoft Windows и IBM AIX ® 5L™.

Локальная установка DB2

При настройке интеграции Domino и DB2 можно выбирать между локальной или удаленной установкой. При выборе локальной установки

оба сервера, и Domino и DB2, будут установлены на одном компьютере. Локальную установку лучше использовать для тестовой среды или для системы с малой производительностью. На [рис. 2.1](#) показана система с локальной установкой.

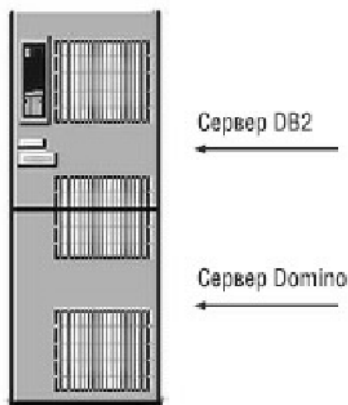


Рис. 2.1. Локальная установка: сервер Domino и сервер DB2 установлены на одном компьютере

Удаленная установка DB2

Для систем с высокой нагрузкой на сервер, скорее всего, лучше будет выполнить удаленную установку. В таком случае сервер Domino и сервер DB2 будут установлены на разных компьютерах. На [рис. 2.2](#) представлен пример удаленной установки.



Рис. 2.2. Удаленная установка: сервер Domino и сервер DB2 установлены на разных компьютерах

Существует два уровня интеграции с DB2. В первом случае сервер Domino и сервер DB2 устанавливаются на одном и том же компьютере, а во втором случае – на разных.

Если вы решили установить оба сервера на одном компьютере, вам необходимо выполнить приведенную ниже процедуру.

1. Установить Domino 7 на сервере Domino.
2. Установить приведенные ниже элементы на сервере DB2:
 - DB2 UDB Enterprise Server Edition версии 8.2.2, включая следующие файлы:
 - DB2 UDB Enterprise Server V8.2;
 - DB2 Universal Database версии 8 с пакетом исправлений 9a.
 - Разблокирующий ключ ограниченного доступа.
3. Установить сервер DB2 Access.

Если вы решили устанавливать оба сервера, Domino и DB2, на разных компьютерах, то вам необходимо выполнить следующую процедуру.

1. Установить Domino 7 на сервере Domino.
2. Установить приведенные ниже элементы на сервере DB2:
 - DB2 UDB Enterprise Server Edition версии 8.2.2, включая следующие файлы:
 - DB2 UDB Enterprise Server V8.2;
 - DB2 Universal Database версии 8 с пакетом исправлений 9a.
 - Разблокирующий ключ ограниченного доступа.
3. Установить DB2 Run-Time Client на сервере Domino.
4. Установить сервер DB2 Access на том же компьютере, что и сервер DB2.

Замечание Убедитесь в том, что вы удалили с компьютера, на котором собираетесь устанавливать сервер DB2 Access, все версии Lotus Notes и Domino, вышедшие раньше версии 7.0. В противном случае система не будет работать. Если есть необходимость сохранить раннюю версию

сервера Lotus Notes/Domino, то переименуйте папку, в которой хранятся файлы этой программы. Убедитесь в том, что `nnotes.dll` не находится в `PATH`.

В данном издании мы обращаем внимание на новые элементы разработки приложений, появившиеся в Domino 7. Поэтому рассматривается только основные моменты процесса установки. Для получения более подробной информации о том, как установить и обслуживать интеграцию Domino и DB2, обратитесь к следующим источникам:

- Lotus Domino Administration 7 Help, "Domino и DB2": ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_admin.nsf/;
- Readme70.pdf (на компакт-диске);
- обучающие ресурсы по DB2, размещенные на Web-сайте IBM: ссылка: <http://www.ibm.com/software/sw-lotus/services/education.nsf/wdocs/educationhomepage>
 - Developing applications with IBM Lotus Domino Enabled For DB2, E-learning Edition: Self Paced (N7D400SP): ссылка: <http://www.ibm.com/software/sw-lotus/services/education.nsf/81F04813DC9CE3C7852566DA004C!>
 - Introducing IBM DB2 Concepts (N7D020): ссылка: <https://education.lotus.com/rw/lewwschd.nsf/5FDC13DD71D11431f>

Совет Во время установки необходимо определить несколько имен пользователей DB2. Имена пользователей DB2 должны соответствовать приведенным ниже правилам присвоения имен и быть написаны заглавными буквами:

- ID (идентификаторы) пользователей в системах, построенных на основе Linux [®] и UNIX [®], могут содержать до 8 символов.
- Имена пользователей в системах, построенных на основе Microsoft Windows, могут содержать до 30 символов. Однако практически у систем Microsoft Windows NT [®], Windows 2000, Windows XP, Windows Server 2003 предел составляет 20 символов.
- Когда не используется клиентская идентификация, клиенты 32-битовых не Windows-систем, подключающиеся к Windows NT, Windows 2000, Windows XP и Windows Server 2003, с именами

пользователей длиннее 8 символов, поддерживаются только в том случае, если имя пользователя и пароль определены в явной форме.

- Имена и ID пользователей не могут:
 - быть словами `USERS`, `ADMINs`, `GUESTs`, `PUBLIC`, `LOCAL` или любыми другими зарезервированными словами SQL (Structured Query Language);
 - начинаться со слов `IBM`, `SQL` или `SYS` ;
 - включать диакритические символы.

2.2 DB2 Access Views

В Lotus Notes и Domino применяется иной принцип хранения данных по сравнению с тем, что используется в реляционных базах данных, таких как DB2. В реляционных базах данных информация упорядочивается и распределяется по строго структурированным таблицам, а в базе данных Domino документы или "записки" (`"notes"`), состоящие из списка полей, сохраняются в формате неструктурированного файла. Такой способ хранения информации усложняет доступ к данным Domino из DB2. Для решения данной проблемы в Domino 7 были введены DB2 Access Views, общие ресурсы, позволяющие определить представление DB2, обращающееся к данным Domino путем систематизации последних в таблицу, которую можно запросить с помощью языка структурированных запросов (SQL). DB2 Access Views являются элементами разработки приложений заднего плана и не видны конечным пользователям.

DB2 Access Views доступны только для баз данных, хранящихся на сервере DB2. Перед созданием DB2 Access View на сервере Domino должна быть активирована DB2 и база данных должна находиться в системе DB2. Если вы создадите локальную реплику базы данных или реплику на сервере Domino 7.0, на котором DB2 не активирована, то DB2 Access View будет находиться в базе данных, однако им нельзя будет воспользоваться или он не будет видимым в Lotus Domino Designer.

Когда в базе данных Lotus Notes включается поддержка DB2, Domino создает схему базы данных DB2 для хранения доступных пользователю представлений Lotus Notes. Имя схемы основывается на имени файла

NSF. Любые созданные вами DB2 Access Views будут также расположены в этой схеме.

Когда вы создаете DB2 Access View, информация копируется в специальную таблицу DB2. После создания и инициализации данных в DB2 Access View Domino автоматически синхронизирует обновления между неструктурированными данными Domino и структурированными данными столбцов в DB2 Access View. Следовательно, обновлять DB2 Access View нет необходимости.

2.2.1 Создание DB2 Access Views

Для создания нового DB2 Access View вам необходимо выполнить приведенные ниже действия. К тому же вы можете обратиться к справке Lotus Domino 7 Designer, которую можно найти по следующему адресу в Интернете: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf ссылка:

1. Откройте базу данных, в которой активирована поддержка DB2 в Lotus Domino Designer 7.0.
2. На панели проектирования выберите *Shared Resources* => DB2 Access views.
3. Щелкните по кнопке New DB2 Access view (Создать новый DB2 Access View). Откроется окно, показанное на рис. 2.3.

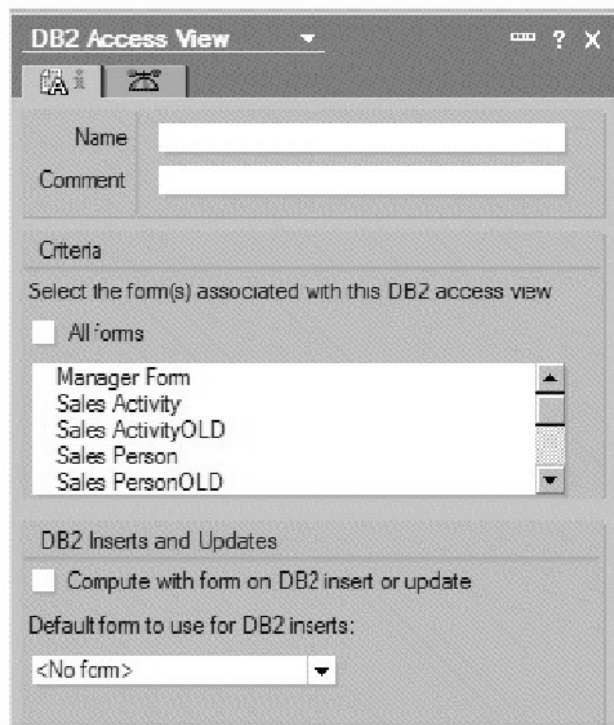


Рис. 2.3. Закладка основных свойств DB2 Access View

4. В окне DB2 Access View вы можете:

- Присвоить имя DB2 Access View.
- Выбрать форму или формы, связанные с DB2 Access View. Вы можете выбрать, какие именно документы включать, базируясь на содержании поля `Form` документа. Можно либо выбрать все формы, либо выбрать некоторые из доступных форм, приведенных в списке. Если у документа отсутствует поле формы, он будет включен в DB2 Access View, только если вы выберете пункт меню `All Forms` (Все формы).
- Параметры `DB2 Inserts and Updates` будут верны только в том случае, если вы выполните операторы `insert` или `update` SQL-выражения из DB2:
 - `Compute with form on DB2 insert or update` (Пересчитывать по форме при операциях DB2 insert и update). Включите эту опцию, если

выбранные формы содержат вычисляемые поля и вы хотите, чтобы формулы были пересчитаны, когда документ создается или обновляется с помощью SQL.

- `Default form to use for DB2 inserts` (Форма по умолчанию для вставок DB2). Пользователи DB2 могут осуществлять операции вставки, обновления и удаления (если у них есть права на это) с данными в DB2 Access View. В Domino 7 можно выбрать только один тип документа (основанный на форме) для создания с помощью SQL-оператора `INSERT`. Документы любого типа (основанные на любой форме) могут быть обновлены или удалены из SQL.

5. Перейдите к закладке `Advanced` (Дополнительные) окна DB2 Access View, как показано на [рис. 2.4](#).

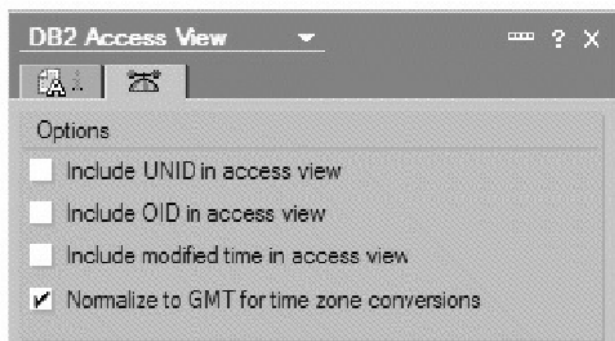


Рис. 2.4. Закладка `Advanced` окна DB2 Access View

Здесь вы можете выбрать приведенные ниже опции:

- `Include UNID in access view` (Включить UNID в access view).

Когда вы установите этот флажок, вы сможете получить доступ к уникальному идентификатору документа Lotus Notes (`UNID`) через DB2 Access View.

- `Include OID in access view` (Включить OID в access view).

OID расшифровывается как Origination Identifier

(идентификатор создания). Если вы выберете эту опцию и выберете ее в SQL-формуле `Query View`, то ссылки документа будут работать, когда элементы вида выбраны и скопированы в виде таблицы.

- `Include modified time in access view` (Включить отметку времени изменения документа в `access view`).

Выберите эту опцию, чтобы включать отметку времени изменения документа в `Notes` для каждой записи в `DB2 Access View`. Обратите внимание на то, что эта отметка времени преобразуется в стандарт `GMT` (Гринвичское время).

- `Normalize to GMT for time zone conversions` (Нормализировать под `GMT` для осуществления преобразований между различными часовыми поясами).

Выберите эту опцию, чтобы стандартизировать все даты и временные отметки в виде `DB2` под `GMT`. Данная опция особенно полезна для распределенных приложений `DB2`, к которым обращаются пользователи, находящиеся в разных часовых поясах.

6. Закройте окно свойств `DB2 Access View`.
7. Нажмите кнопку `Choose Field` (Выбрать поле), после чего на вашем экране появится окно, показанное на [рис. 2.5](#).

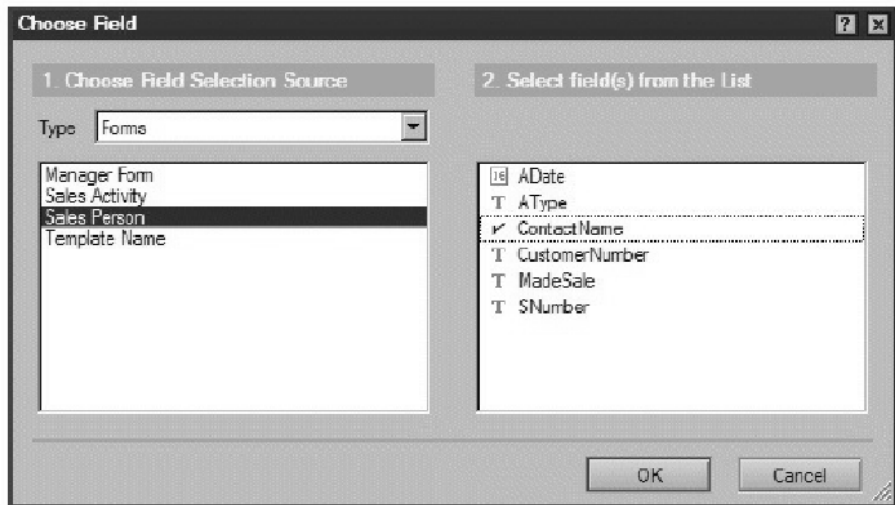


Рис. 2.5. Окно DB2 Access View Choose Field

8. В окне `Choose Field` вы можете:

- Выбрать поля, которые хотите включить из каждой формы.

В Domino 7 DB2 Access View может содержать максимум 84 столбца. Если вам понадобится большее количество данных, придется определить несколько DB2 Access Views.

- Щелкните по кнопке `OK` для подтверждения выбора.

Список выбранных элементов DB2 Access View появится в рабочей области.

9. Нажмите кнопку `Insert Field` (Вставить поле), чтобы включить дополнительные поля.

Возможно, вы захотите добавить поля, находящиеся вне выбранных форм. Например, возьмем поле `Form`. Вы можете вручную добавить его с помощью командной кнопки `Insert Field`. Каждый раз, когда вы используете подобное действие, появляется безымянный элемент DB2 в рабочей области.

10. Дважды щелкните по каждому элементу `Access View Entry`, чтобы отредактировать его свойства. См. [рис. 2.6](#).

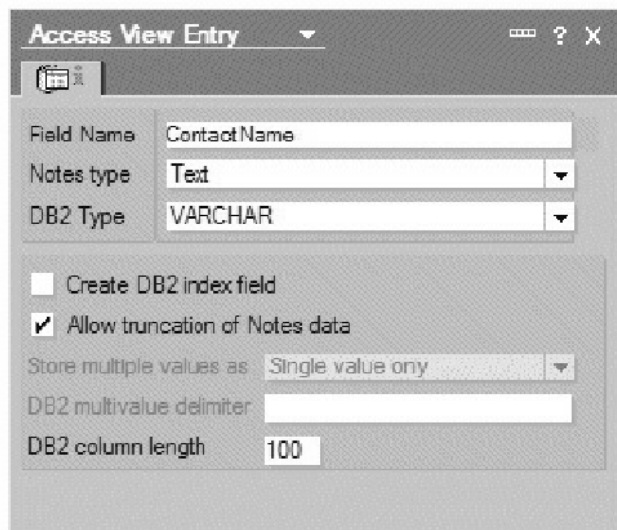


Рис. 2.6. Свойства Access View Entry

Для полей, которые были добавлены с помощью команды `Insert Field`, первые три свойства должны быть заполнены. Для полей, выбранных из форм, соответствующие свойства заполняются автоматически. В этом окне присутствуют следующие поля для заполнения:

- `Field Name` (Название поля).

В это поле необходимо ввести соответствующее имя поля для столбца DB2 в DB2 Access View, поэтому необходимо ввести реальное имя поля, которое появится в записи (в противном случае соответствующее поле найдено не будет и столбец при просмотре в DB2 будет пустым).

Замечание Если использовался метод `Choose Field` при определении полей для DB2 Access View, вы сможете редактировать имена полей в окне свойств. Однако если вы измените название поля таким образом, что оно не будет соответствовать названию поля в документе, соответствующий столбец в DB2 окажется пустым.

- `Notes type` (Тип данных Notes).

Показывает тип данных Lotus Notes для этого поля.

Замечание Следующие типы данных Notes не поддерживаются в DB2 Access Views: *Formula*, *rich text*, *rich text light*.

- DB2 Type (Тип данных DB2).

Показывает тип данных DB2 для данного поля. Lotus Notes присваивает этому полю значение по умолчанию, связанное с типом выбранного поля Notes.

Замечание Если имеется целое число (*integer*), определенное в DB2 Access View, и вы пытаетесь обновить этот столбец с помощью операторов SQL INSERT или UPDATE, используя при этом действительное число (*real*), то перед обновлением DB2 преобразует действительное число в целое, и десятичная часть числа будет отброшена.

Представленные ниже атрибуты значимы для всех элементов DB2 Access View.

- Create DB2 index field (Создать индексированное поле DB2). Этот столбец DB2 индексируется для увеличения скорости поиска по базе данных. Используя приведенную ниже команду, эта опция создает DB2-индекс, основанный на столбце таблицы:

```
> CREATE INDEX ON TABLE x (coly)
```

Использование индексов требует дополнительного пространства на системе хранения и обновления большего количества ресурсов, зато запросы такого типа выполняются намного быстрее, особенно если в TABLE x хранится большое количество строк:

```
> SELECT cola, colb, colc FROM x WHERE coly=<some value>
```

Индексация в DB2 – это расширение возможностей для разработчиков в среде Domino, однако существуют индексы, оптимизирующие выборку данных, которые нельзя создать в Domino Designer. Например, эта команда оптимизирует следующий

запрос:

```
> CREATE INDEX ON TABLE x (cola, colb, colc)
```

В данном примере индекс используется для сортировки данных. При отсутствии индекса, когда таблица увеличивается, эта операция является наиболее ресурсоемкой:

```
> SELECT cola, colb, colc FROM x WHERE cola = <some value> ORI  
cola, colc
```

- **DB2 column length** (Длина столбца DB2). Это свойство определяет длину столбца в DB2 Access View. При выборе значения для данного поля зачастую приходится искать компромисс между противоположными целями. Возможно, вы захотите отобразить максимально возможное количество полей, но помните, что максимальная длина столбца непосредственно влияет на пространство, занимаемое DB2 Access View в базе данных DB2. При преобразовании полей *Notes* необходимо помнить о том, что существует предел максимального количества данных, которые можно сохранить в таблице DAV. Длина всех полей данных вместе с зарезервированным DB2-пространством не может превышать 16384 байт. Если необходимо хранить более 16000 байт данных, то часть текстовых полей можно преобразовать в тип **LONG VARCHAR**. При преобразовании поля в этот тип данных, в таблице сохраняется небольшой "локатор" ("locator"), а текущее поле данных (до 32000 байт на одно поле) сохраняется в виде большого символьного объекта (**Character Large Object, CLOB**). Использование **CLOB** позволяет увеличить емкость DAV (84*32000), однако вы должны использовать тип **LONG VARCHAR**, только если не можете поместить данные "линейно", так как **CLOB** работает медленнее и занимает больше дискового пространства, чем линейное хранение данных. При создании или обновлении DB2 Access View сервер Domino автоматически проверит DB2 Access View на превышение максимальной длины; таким образом, не придется вручную считать длину. Тем не менее, вы можете использовать рекомендации, приведенные в [табл. 2.1](#), для оценки "линейной" длины данных.

Тип	Длина в DAV	Комментарии
VARCHAR	5 + заявленная длина	Например, для хранения VARCHAR (40) потребуется 45 байт
LONG VARCHAR	25	Линейный локатор
DOUBLE	9	Фиксированный размер
TIMESTAMP	11	Фиксированный размер
DATE	5	Фиксированный размер
TIME	4	Фиксированный размер
INTEGER	9	Фиксированный размер

- Allow truncation of Notes data (Позволить усечение данных Notes). Иногда размер поля может быть больше, чем максимальная длина столбца, которую вы определили для элемента DB2 Access View. Когда вы разрешите усечение данных, содержание по достижении максимальной длины будет обрезаться. Если вы не разрешите усечение, то при достижении предела во время заполнения DB2 Access View на консоли сервера появится следующее сообщение: "Access Table field data exceeds defined length".

В Domino 7 параметр Allow truncation of Notes data по умолчанию отключен. В дальнейшем мы настоятельно рекомендуем включить этот параметр для всех элементов DB2 Access View Entry, в которых содержатся текстовые данные. Однако существует одно исключение: если вы планируете делать SQL-запросы на обновление DB2 Access View из DB2, этот параметр дает разрешение на усечение данных до заданной длины, что может привести к потере данных.

Замечание Мы рекомендуем вам выбрать параметр Allow truncation of Notes data для всех элементов Access View.

2.2.2 Экспортирование данных с типом TIMEDATE в DB2 Access View

Тип данных Lotus Notes `TIMEDATE` позволяет хранить дату, время, временную зону, а также индикатор летнего времени, однако временная отметка DB2 позволяет хранить только дату и время. В DB2 считается, что все даты и время локальны по отношению к часовому поясу сервера DB2. Это может привести к неверному отображению времени при экспортировании данных с типом `TIMEDATE` в DB2 Access View или к чтению данных с типом `TIMESTAMP` из DB2 в Query View (вне зависимости от того, откуда они взяты: из DB2 Access View или же из других таблиц DB2).

Существует два варианта сохранения даты и времени в DB2 Access View: локальное (местное) время или время, измененное в соответствии с GMT (стандартизованное). Вы можете изменить способ работы с ними, используя параметр *"Normalize to GMT for time zone conversions"* на закладке Advanced окна свойств Access View Entry.

Замечание Если этот параметр изменен в существующем DB2 Access View, вы должны будете перестроить DB2 Access View.

2.2.3 Размещение диапазонов дат в DB2 Access View

При вставке или обновлении диапазонов дат в документе, расположенном внутри DB2 Access View через SQL, необходимо определить диапазоны дат в полном формате временной отметки DB2, например такой, как `yyyy-mm-dd-hh.mm.ss.subsec` (год-месяц-день-час.минута.секунда.миллисекунда) или `2005-12-12-08.06.30.123456`. Если при выполнении данного этапа произошла ошибка, вставленное значение будет не определено или равно нулю.

2.2.4 Проверка статуса DB2 Access View

При выборе DB2 Access View в Domino Designer показывается список всех DB2 Access View, находящихся в базе данных. Перед каждым DB2 Access View отобразится иконка, показывающая его статус. Данные иконки с соответствующими расшифровками приведены на [рис. 2.7](#).

	New DAV, not yet created or populated in DB2
	New DAV that has been created in DB2 (but not yet populated)
	DAV has been successfully created and populated in DB2
	There is an error in the DAV

Рис. 2.7. Иконки, символизирующие статус каждого DB2 Access View

2.2.5 На что еще обратить внимание при создании DB2 Access View

Вы можете создать сколь угодно много DB2 Access View в базе данных, например по одному на каждую форму. Но может оказаться проще создать один или два больших DB2 Access View и включить в них данные из всех форм, к которым вы хотите иметь доступ. Чтобы ограничить количество столбцов в DB2 Access View, попробуйте использовать одинаковое имя для соответствующих полей в разных формах.

Domino невероятно гибок, когда дело доходит до полей и соответствующих им типов данных. Два документа, основанных на одной и той же форме, в Lotus Notes и Domino могут иметь разные типы данных. Это преимущество в гибкости может вызвать проблемы, например когда вы попытаетесь создать DB2 Access View. DB2 крайне строга по отношению к типам данных. Необходимо будет часто чистить данные в базе данных, а также быть уверенным в том, что содержимое всех полей, использованных в DB2 Access View, обладает одним и тем же типом данных.

Если в документе Lotus Notes будет потеряно какое-либо поле, или

значение этого поля будет отсутствовать, то никаких проблем не возникнет. В DB2 Access View значение такого поля будет просто установлено =NULL, и при любых запросах, касающихся этого поля, будет отображаться пустое значение. Поля с типом *rich text* и *formula* не поддерживаются в DB2 Access View. Если вы включите такие поля в DB2 Access View, то сообщения об ошибке выдаваться не будет, но вместо содержимого поля будет отображаться NULL.

Размеры DB2 Access View и его элементов (таблиц, видов, триггеров и индексов) не включаются в итоговый размер файла базы данных Lotus Notes с поддержкой DB2, для которой оно было создано. Таким образом, если у вас есть множество DB2 Access View, связанных с базой данных Lotus Notes с поддержкой DB2, следует помнить, что они займут такое количество дискового пространства, которое будет превышать дисковое пространство, отведенное под саму базу данных Lotus Notes с поддержкой DB2. Но опасения, что DB2 Access Views будут занимать большее количество дискового пространства, оказались необоснованными. На ранних этапах тестирования в IBM было показано, что даже DB2 Access View с большим количеством столбцов обычно занимает всего лишь от 2% до 3 % от дискового пространства, занимаемого базой данных.

2.2.6 Применение DB2 Access View в рамках приложения ITSO Electronics

В нашем сценарии, приведенном в качестве примера, компании ITSO Electronics требуется использовать новую возможность *Query Views*, к обсуждению которой мы перейдем в п. 2.3, "Query Views". Для того чтобы подготовить базу данных к использованию *Query Views*, нам необходимо создать несколько DB2 Access View. В этом примере мы решили использовать только по одному DB2 Access View на каждую базу данных Lotus Notes приложения ITSO Electronics.

Для начала создадим DB2 Access View для базы данных ITSO Electronics Sales, которую назовем SALESDAV.

В табл. 2.2 представлены выбранные поля.

Таблица 2.2. DB2 Access View SALESDAV в базе данных Sales

Поле, включенное в DB2 Access View	Форма
ANumber	Sales Activity
AType	Sales Activity
ContactName	Sales Activity
CustomerName	Sales Activity
CustomerNumber	Sales Activity
MadeSale	Sales Activity
Product1	Sales Activity
Product2	Sales Activity
Product3	Sales Activity
Product4	Sales Activity, Sales Person
SName	Sales Activity, Sales Person
SNumber	Sales Activity, Sales Person
SPhone	Sales Person
STitle	Sales Person
FORM	Sales Activity, Sales Person

Следующий DB2 Access View, который мы создадим в базе данных ITSO Electronics Customer, будет называться CUSTOMERDAV (см. [табл. 2.3](#)).

Таблица 2.3. DB2 Access View CUSTOMERDAV в

база данных Customer

Поле, включенное в DB2 Access View	Форма
customerAddress	Customer
CustomerDescription	Customer
customerName	Customer
customerNumber	Customer
ownerName	Customer
FORM	Customer

И, наконец, мы создадим DB2 Access View в базе данных ITSO Electronics Products и назовем его CUSTOMERDAV.

Таблица 2.3. DB2 Access View CUSTOMERDAV в базе данных Customer

Поле, включенное в DB2 Access View	Форма
productDescription	Product
SupplierAddress	Supplier
SupplierDescription	Product, Supplier
SupplierName	Supplier
SupplierNumber	Product, Supplier
productName	Product
productNumber	Product
FORM	Product, Supplier

В дальнейшем мы используем эти DB2 Access Views при создании Query View, которое рассмотрено в [п. 2.3, "Query Views"](#).

2.3 Query Views

Query Views – это новый тип видов в базе данных Notes (NSF). Они заполняются путем выполнения запросов, написанных на языке структурированных запросов (SQL). На первый взгляд кажется, что виды Query Views – это всего лишь еще один способ сбора и представления данных. Однако при более пристальном изучении становится очевидно, что Query Views заставляют вас по-новому посмотреть на данные в Lotus Domino. Здесь впервые возможно отображать информацию более чем из одной базы данных Lotus Notes в пределах одного вида. Вы можете использовать приведенные ниже конструкции SQL.

- Оператор Join.

Документы, находящиеся в несвязанных элементах разных DB2 Access View, объединяются в одну строку в Query View. Полностью этот процесс описан в [п. 2.3.2](#), "Создание Query View с помощью оператора Join".

- Оператор Union.

Элементы из различных DB2 Access Views собираются в одно представление. Мы остановимся более подробно на этом аспекте в [п. 2.3.3](#), "Создание Query View с помощью оператора Union".

- Условие Group by.
- Условие Having.

Запросы динамически строятся на основе языка SQL и не занимают места в базе данных. SQL-выражение, в свою очередь, может быть динамически построено с использованием языка формул Lotus Notes. Вы можете свободно использовать такие @-формулы, как @Username и @Today, не опасаясь возникновения проблем с индексированием, которые могли бы возникнуть в стандартных представлениях Lotus Notes. Можно даже управлять содержимым представления с помощью @-формулы взаимодействия пользователя с системой @Prompt. Описанием подобного рода аспектов мы займемся в [п. 2.3.4](#),

"Динамическое создание Query Views с помощью @-формул".

2.3.1 Создание Query View

Query View не способен напрямую отображать данные документа Lotus Domino. Чтобы сделать это, необходимо в первую очередь, создать DB2 Access View, как описано в п. 2.2.1, "Создание DB2 Access Views". Затем можно приступить к созданию Query View, выполнив приведенную ниже последовательность действий.

1. В Domino Designer перейдите к представлениям (Views). Щелкните по кнопке New View. В окне Create View, в разделе Selection conditions выберите By SQL Query (В соответствии с запросом SQL), как это показано на [рис. 2.8](#).

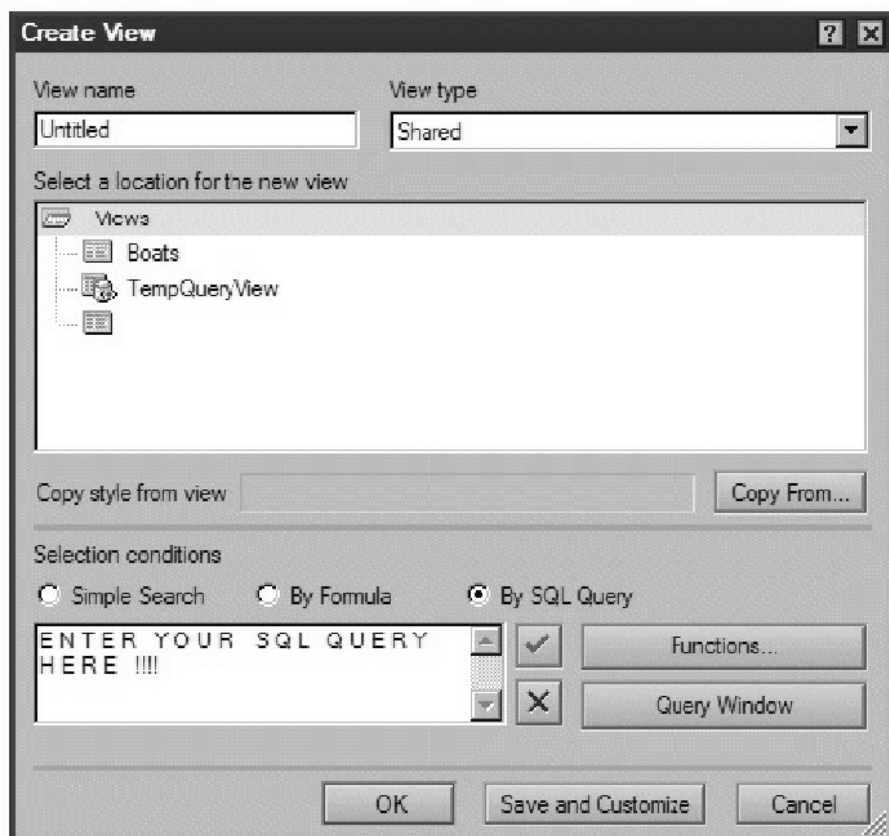


Рис. 2.8. Окно Create View

2. Далее введите SQL-выражение для создаваемого представления. В [п. 2.3.4](#), "Динамическое создание Query Views с помощью @-формул", будет рассказано, как вы можете динамически создавать SQL-выражения, основываясь на языке @-формул. Для того чтобы воспользоваться этой возможностью, статические операторы SQL необходимо заключить в кавычки, как показано в [примере 2.1](#).

"SELECT * FROM SALES.SALESDAV"

Пример 2.1. Операторы SQL должны быть заключены в кавычки

Вы обращаетесь к DB2 Access View в базе данных, используя название схемы DB2 и название DB2 Access View: DB2SCHEMANAME.DAVNAME. В [примере 2.1](#) DB2 Access View было присвоено имя SALES.SALESDAV. Когда вы размещаете базу данных на сервере, название схемы DB2 обычно является частью имени файла без расширения "NSF". В нашем примере SALES.NSF стало SALES. Если такое название схемы DB2 уже используется, то она будет называться SALES2, SALES3 и т. д.

Важно Когда вы создаете SQL-выражение для Query View, точно не известно, будет ли текущее название схемы DB2 окончательным. Для того чтобы быть уверенным в том, что вы ввели правильное название схемы, лучше всего использовать динамические SQL-выражения, речь о которых пойдет в [п. 2.3.4](#), "Динамическое создание Query Views с помощью @-формул".

3. Когда вы написали SQL-выражение, можно определить столбцы представления. Это делается так же, как и в случае с обычными представлениями.
4. Чтобы посмотреть результат, щелкните по кнопке Notes Preview (Предварительный просмотр Notes). Обращаем ваше внимание на то, что кнопка Refresh на панели программирования не позволяет увидеть представление в Domino 7.

Возможно, вы захотите разделить на части ваше SQL-выражение и записать его в несколько строк для удобочитаемости, как показано в

примере 2.2, однако применение такого метода при написании SQL-выражений приведет к появлению сообщений об ошибках.

```
"SELECT  
SALES.SALESDAV.CUSTOMERNAME  
FROM  
SALES.SALESDAV"
```

Пример 2.2. Такой способ написания SQL-выражений вызовет появление сообщений об ошибках

Когда мы работали над созданием приложения ITSO Electronics, то пришли к выводу, что написание SQL-выражений в виде строк, объединенных знаком "+", позволяет достигнуть той же легкости при чтении и что, самое главное, они работают в Lotus Domino 7. См. пример 2.3. Каждая часть SQL-выражения должна быть заключена в кавычки.

```
"SELECT "+  
"SALES.SALESDAV.CUSTOMERNAME "+  
"FROM "+  
"SALES.SALESDAV"
```

Пример 2.3. Этот способ написания SQL-выражений удобочитаем и работает

Язык SQL позволяет использовать упрощенный способ написания SQL-выражений, в котором используются псевдонимы для источников данных. Код программы, представленный в примере 2.3, можно переписать по-другому (см. пример 2.4). Хотя фактически разницы нет, впоследствии этот способ написания SQL-выражений поможет вам понять более сложные примеры.

```
"SELECT "+  
"S.CUSTOMERNAME "+  
"FROM "+  
"SALES.SALESDAV S"
```

Пример 2.4. SQL-выражение, написанное более простым способом с помощью псевдонимов

2.3.2 Создание Query View с помощью оператора Join

В Lotus Domino 7 вы можете соединить данные из различных DB2 Access Views в один Query View. При этом не играет никакой роли, находятся ли DB2 Access Views в одной базе данных или в разных. Каждый элемент представления может показывать связанные данные, взятые из различных документов, что возможно благодаря DB2 Access Views.

На [рис. 2.9](#) показано, как информация из DB2 Access Views ORDERDAV и CUSTOMERDAV объединяется в ResultSet SQL.



Рис. 2.9. Создание Query Views, с использованием оператора Join

В [примере 2.5](#) показано соответствующее SQL-выражение, которое было использовано для этой цели.

```
"SELECT "+
"S.AType, S.Anumber,S.Product1 , C.CUSTOMERNAME, C.OWNERNAME "+
" S.CUSTOMERNUMBER, S.MadeSale "+
"FROM "+
"SALES.SALESDAV S "+
"Left outer JOIN "+
"CUSTOMER.CUSTOMERDAV C "+
"ON "+
"S.CustomerNumber=C.CustomerNumber "+
"WHERE "+
"C.OWNERNAME<>""
```

Пример 2.5. SQL-выражение, использующее оператор Join для создания Query View

На [рис. 2.10](#) показано представление Customer Sales, созданное на основе SQL-выражения, рассмотренного в [примере 2.5](#).

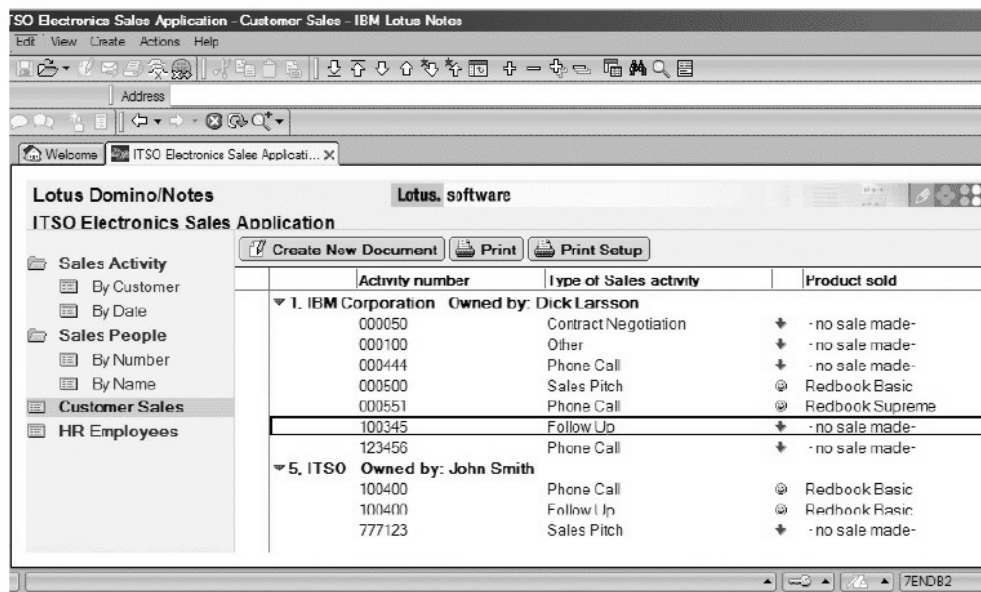


Рис. 2.10. Query View, созданный при помощи оператора Join

2.3.3 Создание Query View с помощью оператора Union

Компания ITSO Electronics настаивает на том, чтобы все ее внешние контакты были объединены в одно представление. При этом информация о поставщиках хранится в базе данных *Products*, а данные покупателей хранятся в базе данных *Customer*. В Lotus Notes и Domino Release 6 данную задачу можно выполнить, лишь сохранив всю информацию в одной базе данных. Однако с появлением Lotus Domino 7 у нас появилась возможность создать Query View с помощью оператора Union в пределах нашего SQL-выражения. На [рис. 2.11](#) показано, как с помощью оператора Union происходит создание Query View из отдельных DB2 Access View.

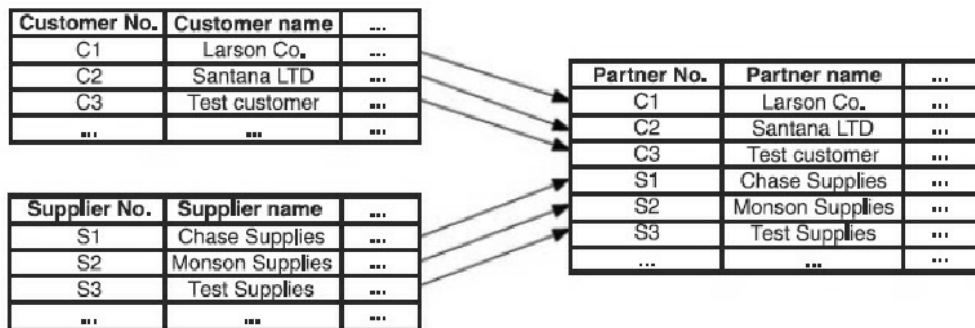


Рис. 2.11. Создание Query Views, с использованием оператора Union

В примере 2.6 показано SQL-выражение, используемое для сбора данных представления, в котором содержатся все внешние контакты компании ITSO Electronics. Для того чтобы оператор Union заработал, необходимо, чтобы каждая формула отбора возвращала одинаковое число столбцов с одинаковыми именами столбцов. К тому же у соответствующих столбцов каждого набора результатов должны быть соответствующие типы полей.

```
"SELECT " +
"C.CUSTOMERNUMBER as NO, C.CUSTOMERNAME as EXTERNALN,
"C.CUSTOMERADDRESS as ADDRESS, C.CUSTOMERDESCRIPTION as
"C.Form as form "+
"FROM "+
"CUSTOMER.CUSTOMERDAV C " +
"WHERE "+
"C.FORM='Customer' "+
"UNION ALL "+
"SELECT "+
"S.SUPPLIERNUMBER as NO, S.SUPPLIERNAME as EXTERNALNAME
"S.SUPPLIERADDRESS as ADDRESS, S.SUPPLIERDESCRIPTION as Des
"S.Form as form "+
"FROM "+
"PRODUCTS.PRODUCTDAV S " +
"WHERE "+
"FORM ='Supplier'"
```

Пример 2.6. SQL-выражение для объединения

На [рис. 2.12](#) показан Query View, отображающий данные поставщиков и покупателей компании ITSO Electronics, располагающиеся в различных базах с использованием SQL-кода, приведенного в [примере 2.6](#).

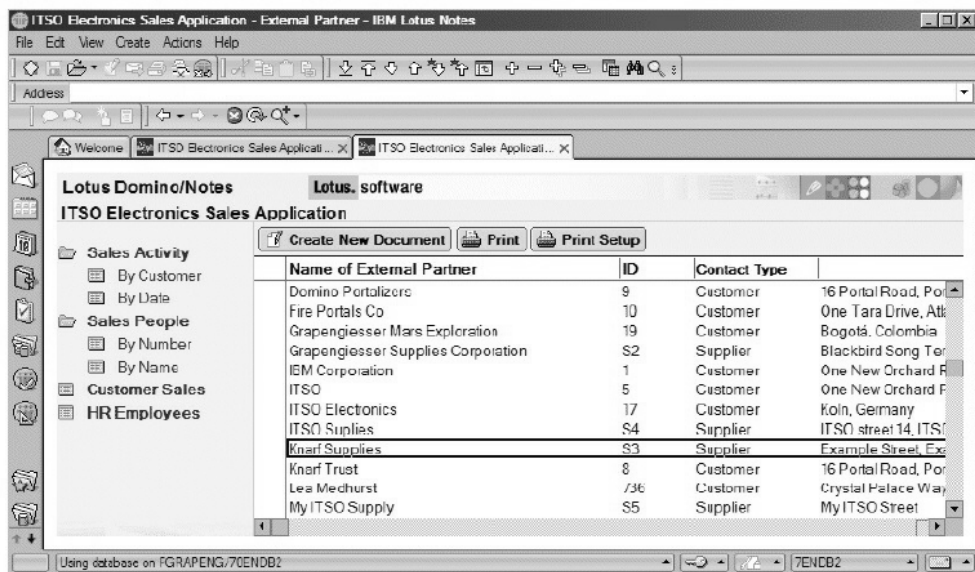


Рис. 2.12. Query View, созданный с помощью объединения

2.3.4 Динамическое создание Query Views с помощью @-формул

Во всех приведенных и использованных до сих пор примерах применялись статические SQL-выражения. Однако во многих случаях лучше создавать SQL-выражения на лету, во время открытия Query View. К счастью, Lotus Domino 7 позволяет составлять текст Query View на языке @-формул.

В [примере 2.7](#) представлено простое SQL-выражение, которое "собирает" все мероприятия по сбыту для покупателя "ITSO" из DB2 Access View SALES.DAV. Для этой цели в нем используется условие `S.CUSTOMERNAME='ITSO'`.

""SELECT "+

```
"S.CUSTOMERNAME, S.ATYPE,S.ANumber , S.MADESALE "+
"FROM "+
"SALES.SALESDAV S "+
"WHERE S.FORM='Sales Activity' AND S.CUSTOMERNAME='ITSO'"
```

Пример 2.7. Статическое SQL-выражение

Вместо этого статического выражения вы можете динамически выбрать название компании при открытии представления. Для достижения этой цели вам необходимо добавить в код команду @Prompt, чтобы пользователь смог ввести название компании, информацию о которой он ищет. См. [пример 2.8](#).

```
_CUSTOMER:=@Prompt([OKCANCELEDIT];"Выбор компании"; "Пожал
название компании, которую хотели бы найти.");";
"SELECT "+
"S.CUSTOMERNAME, S.ATYPE,S.ANumber , S.MADESALE "+
"FROM "+
"SALES.SALESDAV S "+
"WHERE S.FORM='Sales Activity' AND S.CUSTOMERNAME='"+_CUSTC
```

Пример 2.8. Динамически составленное SQL-выражение, использующее язык формул

Теперь каждый раз при запуске этого кода перед пользователем будет появляться запрос на ввод имени покупателя.

Как уже было отмечено в [п. 2.3.1](#), "Создание Query View", вы не можете быть уверены в том, что название схемы SQL для базы данных останется тем же после того, как она будет скопирована или продублирована на другом сервере. Поэтому рекомендуется избегать статических ссылок на схему базы данных в ваших SQL-выражениях. Способ обхождения этого ограничения состоит в использовании функции @DB2Schema которая была внедрена в Lotus Notes и Domino 7. Описание данной функции см. в [п. 4.5](#), "Дополнения к языку формул". Чтобы получить SQL DB2Schema для текущей базы данных, необходимо использовать команду @DB2Schema вместе с функцией @DBName. Если поместить эти команды в SQL-выражение из [примера 2.7](#), мы получим код, похожий на тот, что представлен в [примере 2.9](#).

```

_Schema:= @DB2Schema(@DBname);
"SELECT "+
"S.CUSTOMERNAME, S.ATYPE,S.ANumber , S.MADESALE "+
"FROM "+
_Schema +".SALESDAV S "+
"WHERE S.FORM='Sales Activity' AND S.CUSTOMERNAME='ITSO'"

```

Пример 2.9. Динамически вычисляемая схема текущей базы данных

Также может возникнуть необходимость написать SQL-выражение, позволяющее получить доступ к DB2 Access View в другой базе данных. Если предположить, что эта база данных находится в той же папке, что и текущая, то SQL-выражение, основанное на [примере 2.7](#), будет похоже на решение, представленное в [примере 2.10](#).

```

_pathOnly:=@LeftBack(@Subset(@DbName;-1);"");_path:=@If(_pathOnly='
pathOnly+"\\");_Server:=@Subset(@DbName;1);_Schema:= @DB2Schema(_S
Trim(_path+"Sales.nsf"));
"SELECT "+ "S.CUSTOMERNAME, S.ATYPE,S.AN
MADESALE "+ "FROM "+_Schema +".SALESDAV S "+ "WHERE S.FORM=

```

Пример 2.10. Динамическая ссылка на базу данных, находящуюся в той же папке, что и сервер Lotus Domino

Если при создании SQL-выражения вы не будете уверены в том, по какому пути находится база данных, к которой необходимо получить доступ, можно воспользоваться одной из приведенных ниже команд для получения информации о пути интересующей вас базы данных из конфигурационных документов:

- @DBLookup
- @Environment
- @GetProfileField

2.3.5 Открытие документов из Query Views

В классическом представлении Lotus Notes каждая запись связана с документом напрямую. Таким образом, вы можете открывать, копировать и удалять документы из каждого представления. Однако для

Query View, основанном на SQL-операторах, так бывает не всегда. Перед тем как вы сможете открывать, копировать и удалять какие-либо данные из Query View, необходимо выполнить ряд определенных условий. Строка Query View будет соединена с документом, находящимся в базе данных Notes, если вы включите столбец #NOTEID из DB2 Access View в ваш запрос и этот #NOTEID будет являться корректным Note ID в открытой базе данных Notes.

Например, в нашем исходном Query View, приведенном в качестве примера, мы выполнили следующий запрос:

```
SELECT * FROM SALES.SALESDAV
```

По сути этот запрос выбирает столбец #NOTEID из DB2 Access View, следовательно, создается связь между строками Query View и связанными с ними документами Notes. Приведенный ниже запрос не выбирает #NOTEID из DB2 Access View, поэтому строки в этом Query View не будут связаны с соответствующими документами и сгенерируется ошибка, если вы попытаетесь ими манипулировать (например, дважды щелкнете левой кнопкой мыши):

```
> SELECT CUSTOMERNAME FROM SALES.SALESDAV
```

Если вы выбираете ограниченное число столбцов из DB2 Access View, вам следует отдельно выбрать столбец #NOTEID, например:

```
> SELECT CUSTOMERNAME, #NOTEID FROM SALES.SALESDAV
```

Если вы выполняете операцию JOIN с несколькими DB2 Access Views, убедитесь в том, что вы точно выбрали столбец #NOTEID, который является корректным Note ID (идентификатором записи) в текущей базе данных Notes.

Простейшую связь между видом и документом Notes представляет собой #NOTEID. Тем не менее, некоторые функции, такие как копирование или создание ссылки на документ, требуют OID (идентификатор автора документа). Если вашему приложению требуется такого рода функциональность (или же вы хотите, чтобы взаимодействие с Query View было больше похоже на

взаимодействие с обычными представлениями), включите опцию экспортирования #OID в DB2 Access View (ее можно включить на дополнительной закладке окна свойств DB2 Access View), а также включите в ваши запросы #OID.

Обратите внимание на то, что при выборе данных из разных DB2 Access View, необходимо конкретно указать, из каких DB2 Access View выбирать соответствующие столбцы, например:

```
SELECT S.AType, S.Anumber,S.Product1 , C.CUSTOMERNAME, C.OWNI
C.#OID,
S.CUSTOMERNUMBER, S.MadeSale FROM SALES.SALES DAV S Left on
CUSTOMER DAV C
ON S.CustomerNumber=C.CustomerNumber WHERE C.OWNERNAME<>'
```

2.3.6 Открытие Query View с помощью Web-браузеров

В начальном релизе Lotus Domino 7.0, к сожалению, функция просмотра Query View через Web-браузеры недоступна. Корпорация IBM планирует внедрить поддержку Web-браузеров в последующих релизах Lotus Domino. Для получения дополнительной информации, касающейся обновлений, которые позволяют обеспечить поддержку Web-браузеров, посетите Web-сайт IBM с документацией по продуктам линейки Lotus: ссылка: <http://www.lotus.com/ldd/doc>.

2.3.7 Query View и LotusScript

Несмотря на то, что Query View позволяет открывать документы в представлении, доступ к ним можно получить и с помощью LotusScript (см. п. 2.3.5, "Открытие документов из Query Views"). В нашем случае подойдет простой скрипт, приведенный в примере 2.11. Если вы не можете открыть документы из Query View, то скрипт вызовет ошибку времени выполнения "Object variable not set".

```
Sub Click(Source As Button)
Dim session As New NotesSession
Dim db As NotesDatabase
```

```
Dim view As NotesView
Dim doc As notesdocument
Dim Item As notesitem
Set db=session.currentdatabase
Set view =db.Getview("<ПРЕДСТАВЛЕНИЕ>")
Set doc=view.GetFirstDocument ' <=====
Do While Not doc Is Nothing
Set Item =doc.GetFirstItem("<ЭЛЕМЕНТ В БАЗЕ ДАННЫХ>")
Print(item.text)
Set doc=view.GetNextDocument(doc)
Loop
End Sub
```

Пример 2.11. Простой LotusScript, организующий циклическое прохождение через представление

Существует еще один способ получить доступ к данным в Query Views, не позволяющих открыть документы с помощью LotusScript. Доступ к таким представлениям необходимо осуществлять через класс View Entry, как показано в [примере 2.12](#).

```
Sub Click(Source As Button)
Dim session As New notessession
Dim db As NotesDatabase
Dim view As NotesView
Dim doc As notesdocument
Dim Item As notesitem
Dim entry As NotesViewEntry
Set db=session.currentdatabase
Set view =db.Getview("<Имя Вашего представления>")
Set Entry = View.GetEntryByKey( "<Имя элемента представления>", True)
Msgbox(entry.ColumnValues(1))
End Sub
```

Пример 2.12. Простой скрипт, позволяющий получить доступ к элементам представления

2.3.8 Внедрение Query Views в приложение ITSO Electronics

Как и любое другое представление, вы можете внедрить Query Views в форму, подформу, страницу или в документ. В нашем приложении ITSO Electronics в качестве примера мы выполнили внедрение представления Customer Sales Query View в форму Sales Person. Пользователи, обращающиеся к документу Sales Person, теперь могут без труда увидеть информацию из представления Customer Sales, не закрывая документ (см. [рис. 2.13](#)). Для получения дополнительной информации о создании внедренных представлений, обратитесь к справке Lotus Domino 7 Designer, доступной по следующему адресу: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf

Lotus software

Sales Person

Name: Dick Larsson

Employee Number: 111888

Job Position: Sales Person

Phone Number: 617-555-8888

Region: South East

Sales Activity Number	Sales Activity Type
▼ IBM Corporation	
▼ Owned by: Dick Larsson	
000500	Sales Pitch
100345	Follow Up
123456	Phone Call
000100	Other
000050	Contract Negotiation
000551	Phone Call
000444	Phone Call

Рис. 2.13. Query View, внедренный в форму Sales Person приложения ITSO Electronics

2.3.9 Подробный анализ Query Views

В этом разделе приведены дополнительные аспекты, касающиеся Query View.

Разделение списковых элементов

В DB2 Access Views текстовые списки хранятся в виде текстовой строки, и поскольку Query View считывает данные из DB2 напрямую, именно эта текстовая строка и возвращается в Query View. Таким образом, если у вас есть списковый элемент, который вы хотите разделить в вашем Query View, необходимо использовать функцию `@Explode` в столбце, содержащем несколько значений. Чтобы убедиться в том, что представление способно работать с несколькими строками данных, в окне свойств View $\Rightarrow$ выберите закладку Style, после чего установите параметры Row Height и Shrink Rows to Content в соответствии с вашими данными.

Обновление Query Views

Учитывая то, что Query Views можно построить из комбинаций данных Notes и интегрированных данных, обновление Query View зависит от содержания добавленных или измененных данных и от того, как эти данные были добавлены.

Установка максимального числа строк в запросах SQL

По умолчанию максимальное количество строк, возвращаемых по запросу SQL, равно 500. Существует два способа изменения этой величины:

- Параметр `DB2QueryViewRowLimit=величина` в файле `NOTES.INI`.

Установите значение величины, равное нулю, если хотите снять ограничение с количества возвращаемых строк.

- В закладке Options окна свойств View выберите параметр

Maximum rows returned by a SQL query
(Максимальное количество строк, возвращаемое по SQL-запросу)
и задайте количество строк (см. [рис. 2.14](#)).

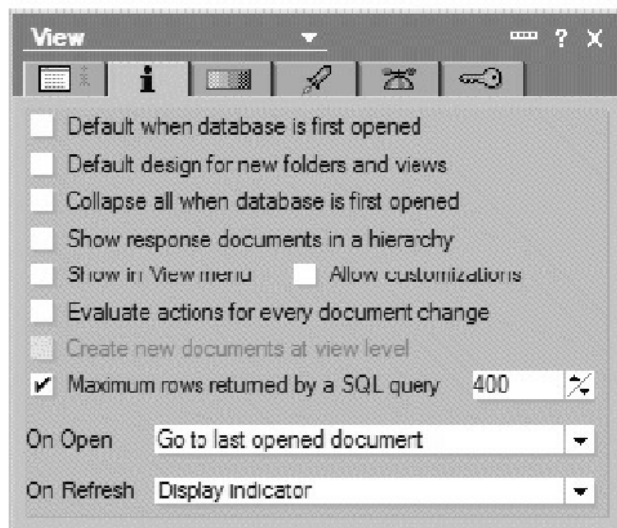


Рис. 2.14. Закладка Options в окне свойств View

2.4 Работа с интегрированными данными DB2

До этого пункта мы занимались обсуждением Query Views, отображающих данные Lotus Domino путем выборки из DB2 Access View. Однако можно создать Query Views, способные отображать любые данные, видимые базе данных DB2, используя DB2 federation – мощный компонент интеграционной архитектуры DB2 и Domino. Интегрированные данные – это данные, которые хранятся в базах данных, не являющихся базами данных Lotus Domino (например, Oracle, Microsoft SQL или другие базы данных DB2). Их можно сделать видимыми для Lotus Domino, изменив конфигурацию базы данных Lotus Domino, включив в ней поддержку интеграции. Впоследствии вы можете создать Query Views, отображающие только данные Lotus Domino, либо только интегрированные данные, либо и то, и другое вместе.

2.4.1 Включение поддержки интеграции для базы

ДАННЫХ

При изменении конфигурации базы данных Lotus Domino в DB2 с целью включения в ней поддержки интеграции, позволяющей получать доступ к объектам, находящимся вне текущей базы данных (база данных определена в закладке DB2 серверного документа, находящегося в директории Domino), вы можете создавать так называемые "интегрированные объекты" (federated objects). Эти интегрированные объекты DB2 позволяют не только получать доступ к другим таблицам DB2, которые находятся в других базах данных DB2, но также и к другим системам управления базами данных (СУБД).

В приложении ITSO Electronics нам необходимо создать новый интегрированный Query View, который будет обращаться к информации отдельной базы данных DB2, принадлежащей отделу кадров и содержащей данные о персонале. В приведенной ниже последовательности действий показано, как изменить конфигурацию сервера DB2 для включения поддержки интеграции.

1. Установите переменную окружения DB2_ALLOW_SETAUTH_WITH_REMOTECONNECT, чтобы Query View смог работать. Используйте следующие путь и команду:

```
C:\Program Files\IBM\SQLLIB\BIN>db2set -i DB2 DB2_ALLOW_SE
```

где -i определяет имя вашего экземпляра DB2 (который обычно, как и в нашем сценарии, называется DB2).

2. Перезапустите DB2 для того, чтобы переменная окружения начала работать. Обратите внимание на то, что с помощью приведенного ниже пути и команды можно обновить конфигурацию DB2 Database Manager, для того чтобы активировать интеграцию в DB2:

```
C:\Program Files\IBM\SQLLIB\BIN>db2 update dbm cfg using FEDER/
```

3. Подключитесь к вашей базе данных Lotus Domino в DB2 Control Center (центре управления DB2). Убедитесь в том, что в базе

данных Lotus Domino включена поддержка интеграции. Для этого щелкните правой кнопкой мыши по вашему экземпляру DB2 и выберите из контекстного меню *Configure Parameters* (Изменить параметры). Откроется окно DBM Configuration (см. [рис. 2.15](#)). Прокрутите это окно вниз до раздела *Environment*. Найдите ключевое слово **FEDERATED** и убедитесь, что в поле *Value* стоит значение **Yes**. Если же в поле *Value* стоит **No**, однако в поле *Pending Value* стоит **Yes**, и вы выполнили команду `db2set`, которую мы рассматривали на этапе 1, необходимо перезапустить DB2, чтобы в поле *Value* ключевого слова **FEDERATED** появилось слово **Yes**.

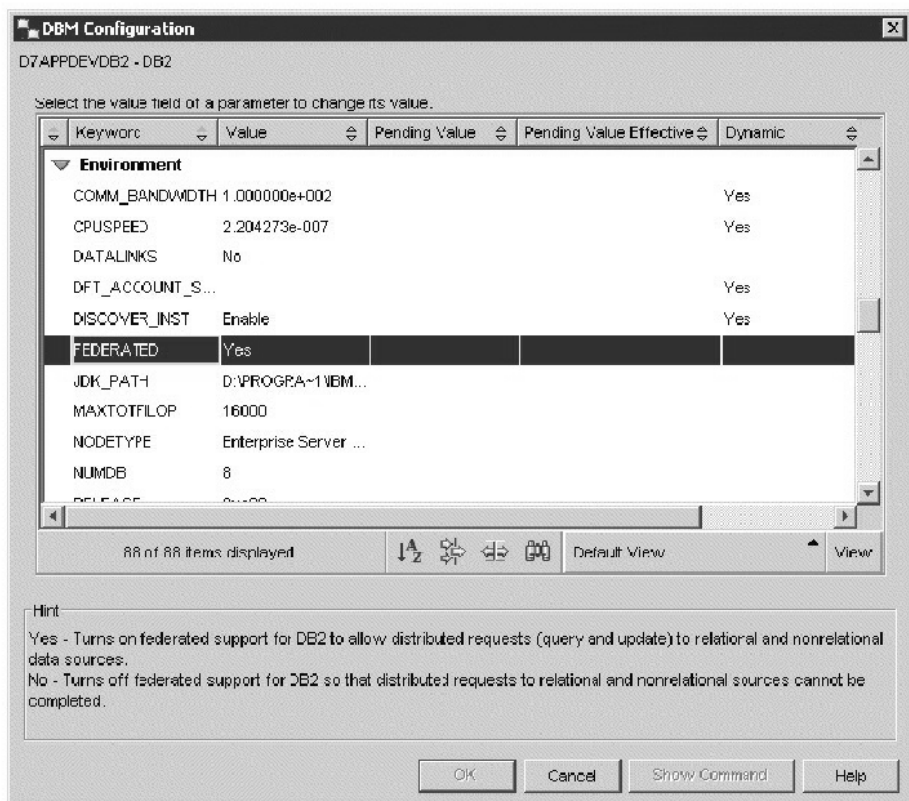


Рис. 2.15. Окно DB2 Control Center: DBM Configuration

- Для того чтобы создать интеграционную оболочку в DB2 для внешней базы данных DB2 (в нашем сценарии это база данных DB2 отдела кадров), необходимо вернуться в DB2 Control Center и

для базы данных Lotus Domino перейти к папке *Federated Database Objects*. Далее щелкните по *Create New Wrapper* (Создание новой оболочки), см. [рис. 2.16](#).

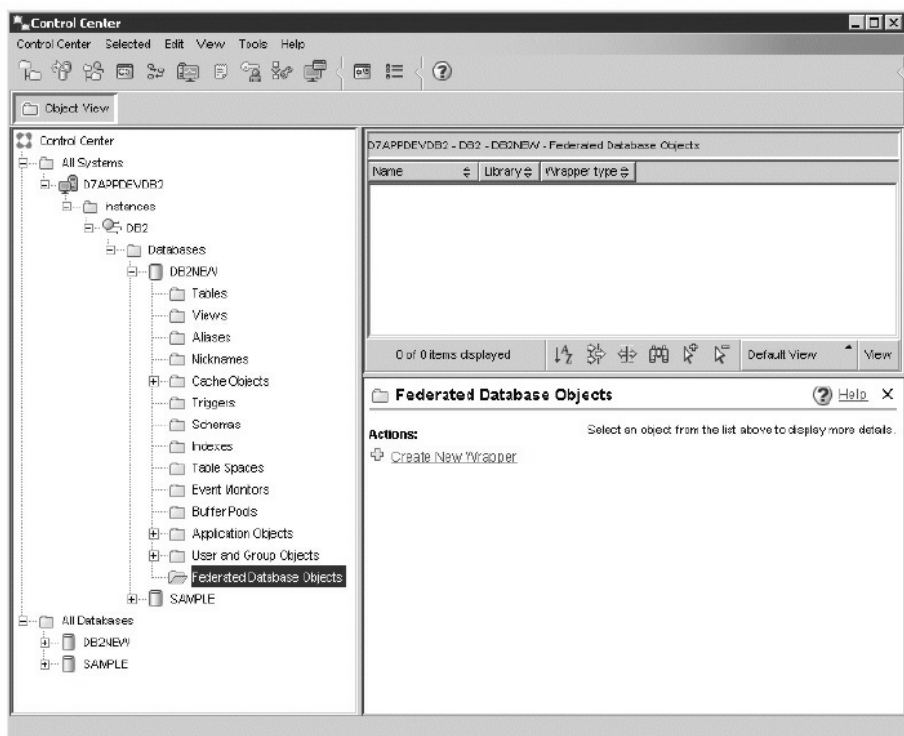


Рис. 2.16. Папка Federated Database Objects в DB2 Control Center

5. В результате откроется окно *Create Wrapper* (см. [рис. 2.17](#)). На закладке *Wrapper* из выпадающего списка *Data source* выберите источник данных, в качестве которого в нашем сценарии мы будем использовать DB2 UDB. Введите название оболочки в поле *Wrapper name*; в нашем случае мы использовали DB2WRAP. Щелкните по кнопке *OK*.

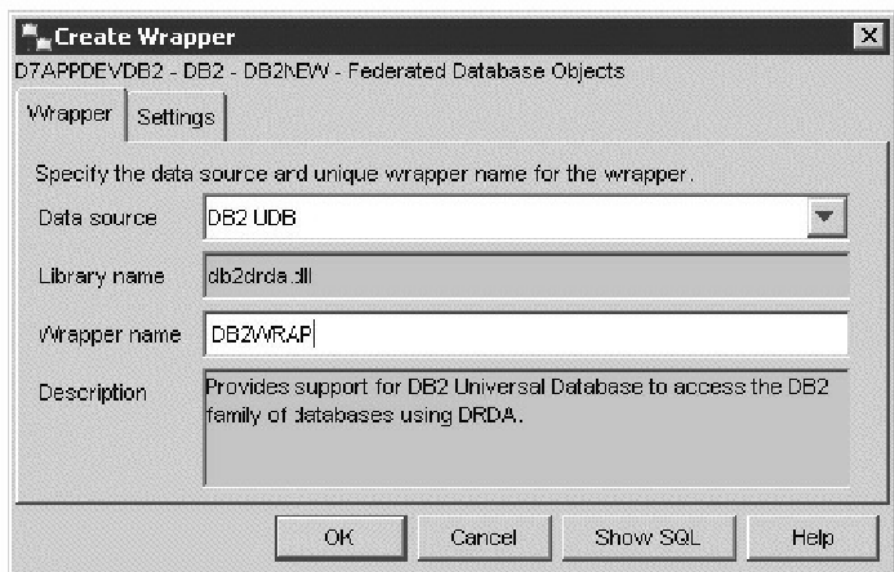


Рис. 2.17. Окно Create Wrapper

Замечание Для получения дополнительной информации, касающейся оболочек, а также включения поддержки интегрированных данных в DB2, посетите Информационный центр DB2, который находится по следующему адресу: ссылка: <http://publib.boulder.ibm.com/infocenter/db2help/index.jsp>

- После того как вы создадите оболочку, она появится в папке *Federated Database Objects* в DB2 Control Center (рис. 2.18). Выберите папку *Server Definitions* (Описания сервера) и щелкните по командной кнопке *Create New Server Definition* (Создать новое описание сервера). Откроется окно *Create Server Definitions*, представленное на рис. 2.19.

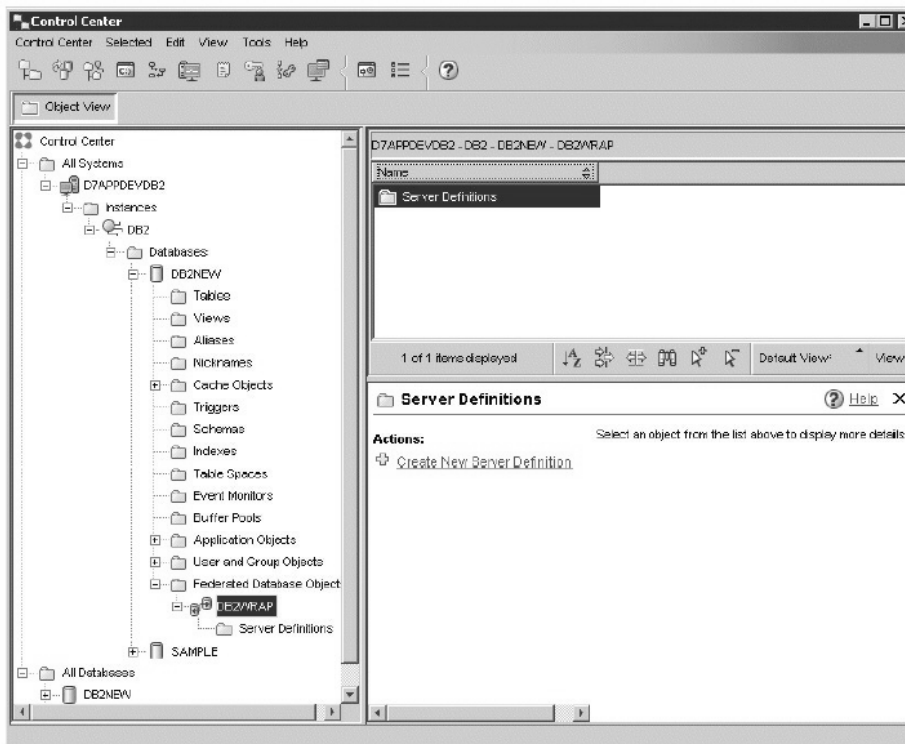


Рис. 2.18. Папка Server Definitions в DB2 Control Center

7. В окне **Create Server Definitions** щелкните по кнопке **Discover** (Найти). Появится список объектов. Выберите подходящий объект, которым в нашем сценарии является база данных **SAMPLE** (так называется база данных DB2 отдела кадров), см. рис. 2.19.

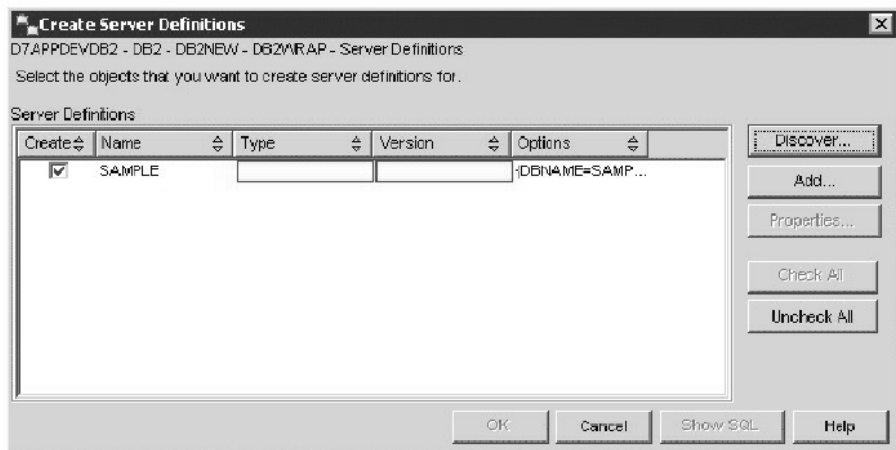


Рис. 2.19. Окно Create Server Definitions: кнопка Discover

- Щелкните по кнопке **Properties** (Свойства), откроется окно **Server Definition Properties** (Свойства описания сервера), которое показано на [рис. 2.20](#). Введите соответствующие значения во все поля закладки **Server Definition**, которая представлена на [рис. 2.20](#). Все введенные в эти поля данные соответствуют нашему сценарию. Для поля **Type** (Тип) из выпадающего списка выберите **DB2/UDB**. Для поля **Version** (Версия) из выпадающего списка выберите **8.2**, а в полях **User ID** (Идентификатор пользователя) и **Password** (Пароль) используйте соответствующие данные пользователя **DB2** (или администратора **DB2**), у которого есть доступ к внешней базе данных **DB2** (в нашем случае это база данных **DB2** отдела кадров).

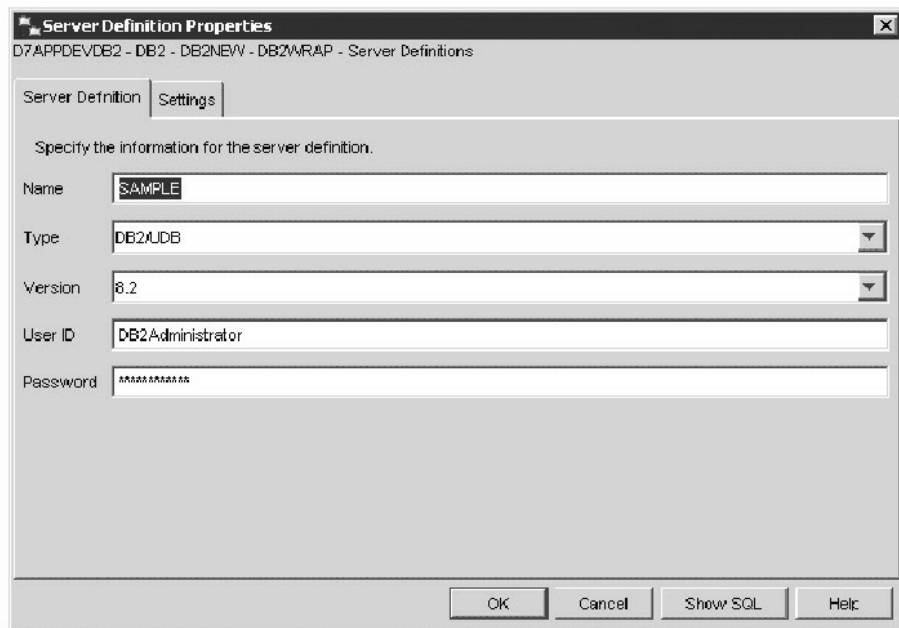


Рис. 2.20. Окно Server Definition Properties

9. Перейдите к закладке *Settings* (Параметры) (см. [рис. 2.21](#)) и введите название базы данных DB2 в поле *Value* (Значение) для параметра *DBNAME*. В нашем сценарии это название базы данных отдела кадров – *SAMPLE*. Кнопка *OK* станет доступной. Щелкните по ней.

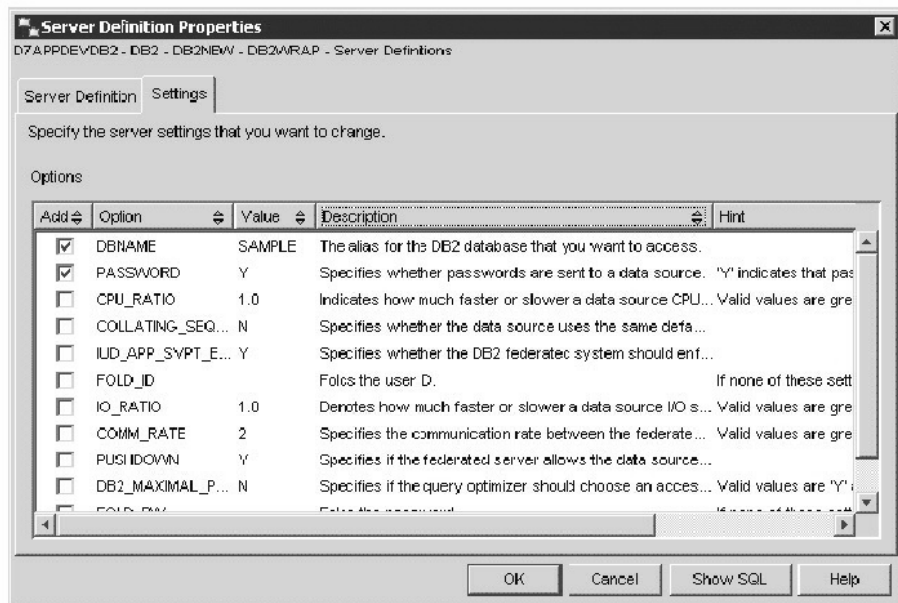


Рис. 2.21. Окно Server Definitions Properties: закладка Settings

10. Теперь нам надо добавить привязку пользователей к этой интеграционной оболочке, присваивая пользователя сервера Lotus Domino к имени и паролю пользователя удаленного соединения. В нашем сценарии в качестве пользователя сервера Lotus Domino мы взяли DB2DOMINO. В DB2 Control Center под созданным описанием сервера для базы данных SAMPLE щелкните по кнопке Create New User Mapping (Создать новую привязку пользователя), как показано на [рис. 2.22](#).

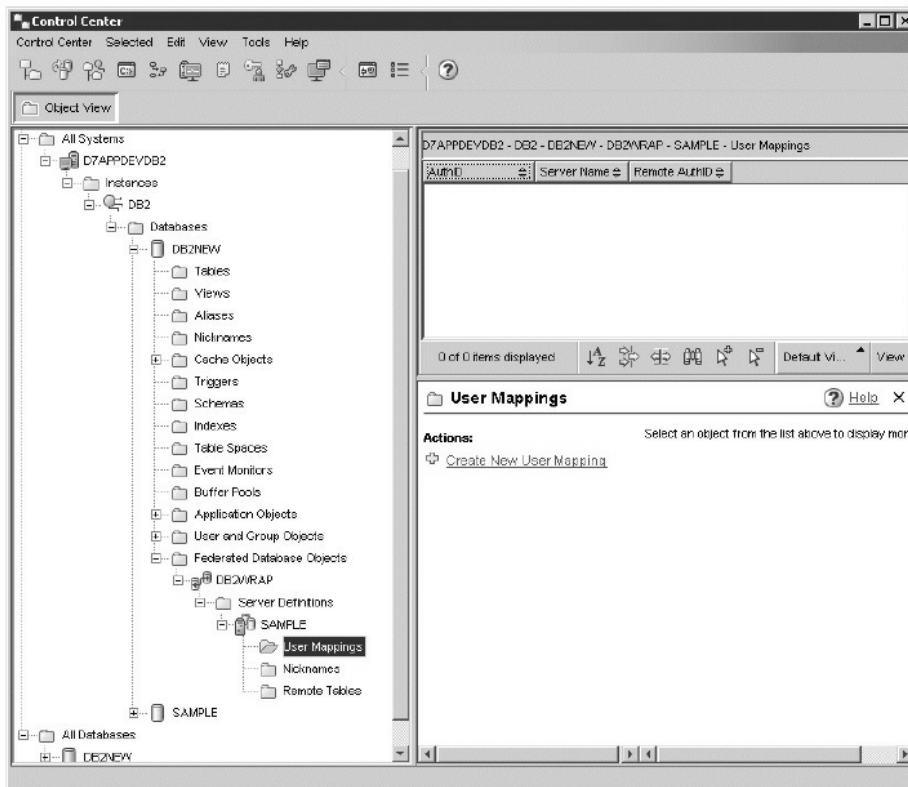


Рис. 2.22. Окно DB2 Control Center: User Mappings

11. Откроется окно **Create New User Mapping** (рис. 2.23). На закладке **Users** (Пользователи) выберите один или несколько идентификаторов (ID) локальных пользователей, которые необходимо связать с удаленным идентификатором. В нашем сценарии мы выбрали ID пользователя **DB2DOMINO**.

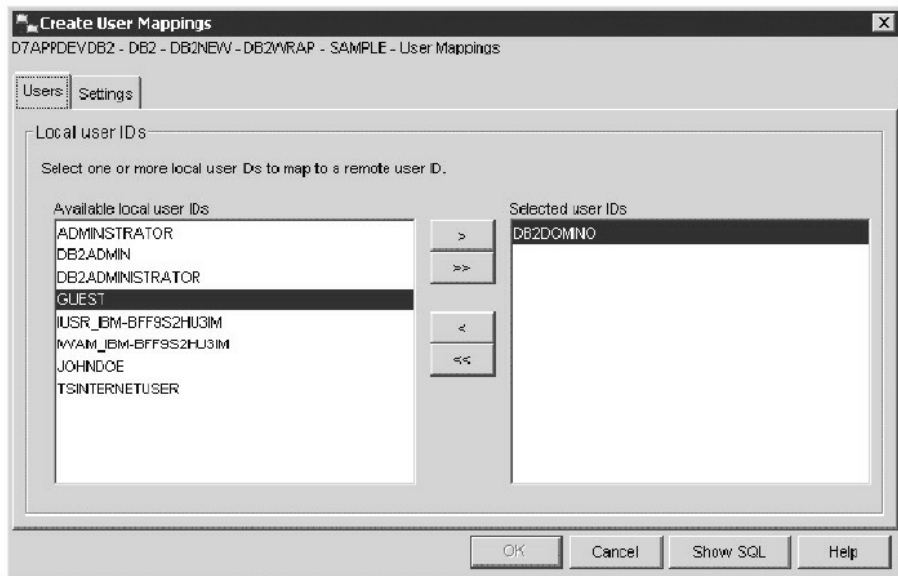


Рис. 2.23. Окно Create User Mappings

12. Перейдите к закладке `Settings` и введите соответствующие значения в столбцы `Value` параметров `REMOTE_AUTHID` и `REMOTE_PASSWORD`. В нашем сценарии мы использовали значения `DB2DOMINO` и `DB2DOMINO` соответственно, для того чтобы появились параметры, представленные на [рис. 2.24](#). (На нашем сервере DB2 также есть ID пользователя `DB2DOMINO`, мы применяли его для реализации привязки пользователя.) Тем не менее, привязка между ID пользователя, с помощью которого Domino подключается к DB2 (в нашем примере в качестве него использован `DB2DOMINO`), и ID пользователя, который зарегистрирован на удаленном компьютере DB2 (параметры `REMOTE_AUTHID` и `REMOTE_PASSWORD`), может быть сделана с использованием идентификаторов пользователей). После этого станет доступной кнопка `OK`. Щелкните по ней.

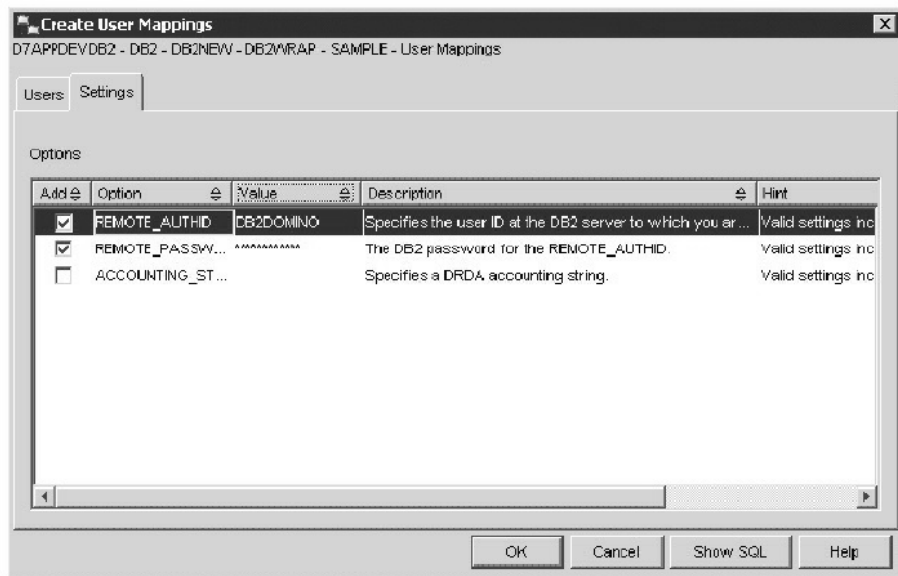


Рис. 2.24. Окно Create User Mappings: закладка Settings

Замечание Если вы столкнетесь с какими-либо проблемами при установке и настройке привязки пользователей, вернитесь в DB2 Control Center, выберите базу данных, к которой хотите подключиться, щелкните по ней правой кнопкой мыши из контекстного меню, выберите пункт **Connect** (Подключиться), после чего введите ID пользователя и пароль. В приложении, использованном в нашем сценарии, мы применяли ID пользователя базы данных DB2DOMINO и его пароль.

13. В DB2 Control Center создайте псевдоним, связывающий локальный объект с внешним объектом. Псевдоним используется в SQL-выражении *Query View*. Выберите папку *Nicknames* (Псевдонимы) и щелкните по командной кнопке *Create New Nickname* (Создать новый псевдоним), как показано на рис. 2.25.

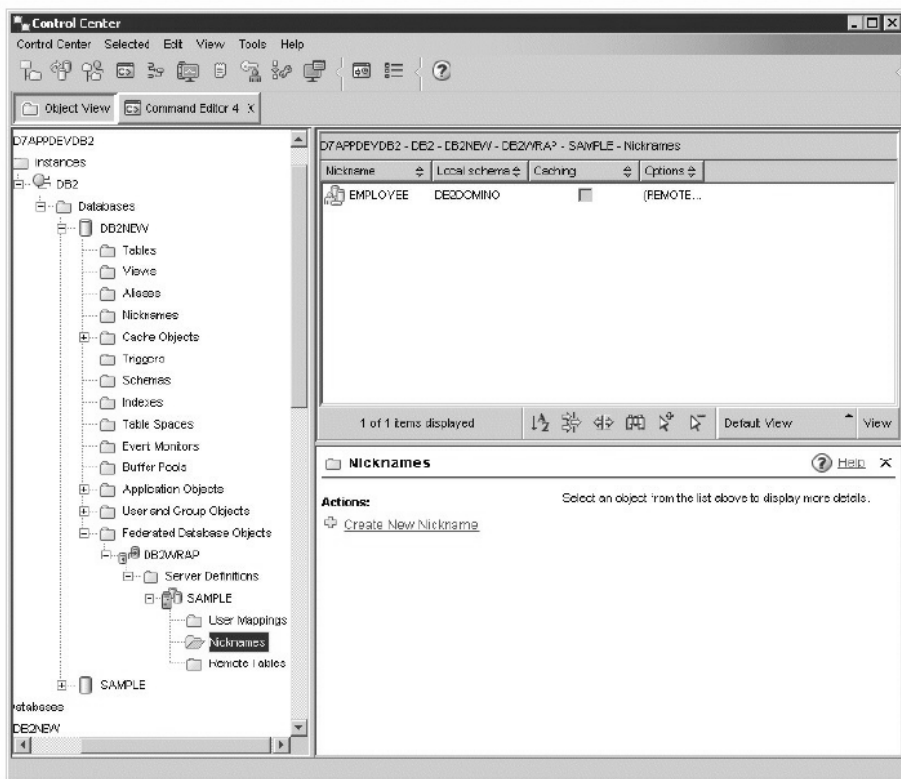


Рис. 2.25. Окно DB2 Control Center: Create New Nickname

14. Откроется окно *Create Nicknames* (Создание псевдонимов). Щелкните по кнопке *Discover*, после чего откроется окно *Discover*, которое представлено на рис. 2.26. Щелкните по кнопке *OK*.

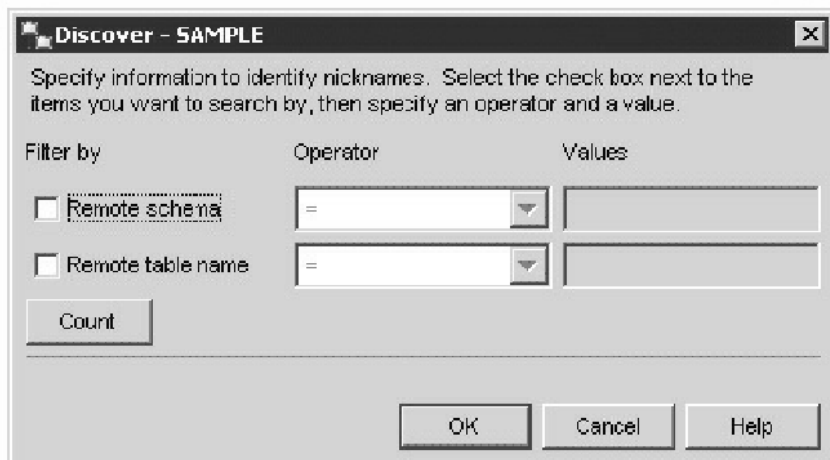


Рис. 2.26. Create Nicknames: окно Discover

15. Затем окно *Create Nicknames* заполнится объектами, которым можно присвоить псевдонимы. В приложении ITSO Electronics для достижения поставленных целей необходимо нажать на кнопку *Uncheck All* (Снять все), а затем поставить флажок в столбце *Create* (Создать) для псевдонима *EMPLOYEE*, как видно из рис. 2.27. Щелкните по кнопке *OK*.

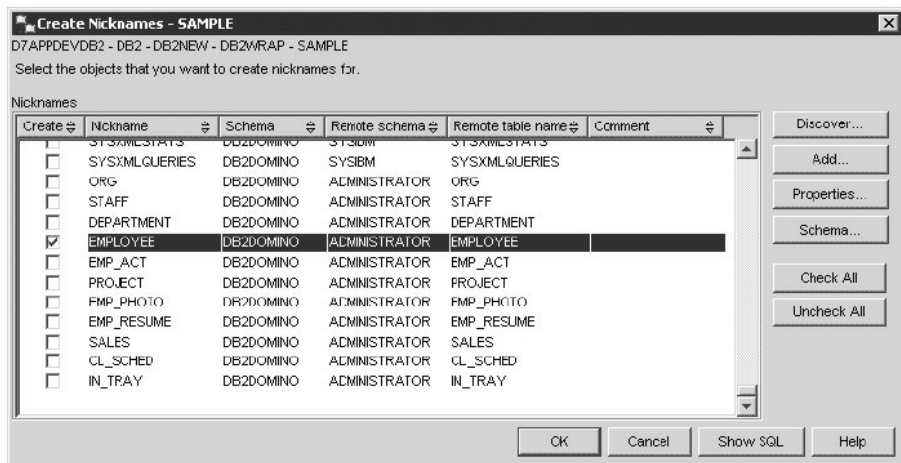


Рис. 2.27. Окно Create Nicknames

16. После этого будет создан псевдоним *EMPLOYEE*, в этом можно убедиться в *DB2 Command Center*. Дважды щелкните по

псевдониму, чтобы отобразить интегрированную таблицу EMPLOYEE, представленную на рис. 2.28.

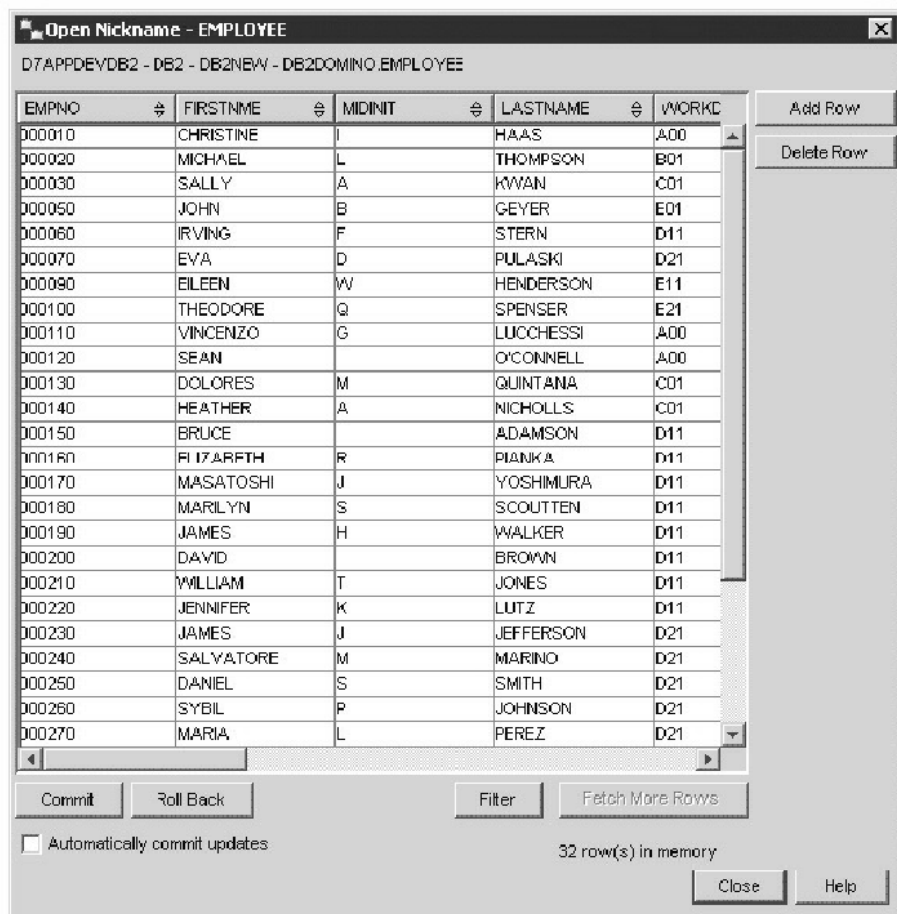


Рис. 2.28. Open Nickname: окно EMPLOYEE

17. Завершающий этап состоит в присвоении привилегий объекту псевдонима, необходимость которых объясняется тем, что из всех пользователей DB2 нужно выбрать тех, которым вы хотите разрешить доступ к интегрированной таблице. В нашем сценарии мы хотим предоставить доступ пользователю с идентификатором MANNYSANTANNA. Чтобы завершить этот этап, выполните приведенную ниже последовательность действий из командной строки или из DB2 Command Editor (Редактор команд DB2). Войдите в вашу базу данных в качестве пользователя сервера Lotus Domino или же в качестве другого администратора. Затем

выполните одно из приведенных ниже действий:

- из командной строки выполните приведенную ниже команду, чтобы предоставить привилегии доступа какому-либо пользователю, в данном случае мы присвоили эти привилегии пользователю MANNYSANTANA:

GRANT SELECT ON DB2DOMINO.EMPLOYEE TO MANN

- выполните эту же команду из DB2 Command Editor (см. [рис. 2.29](#)).

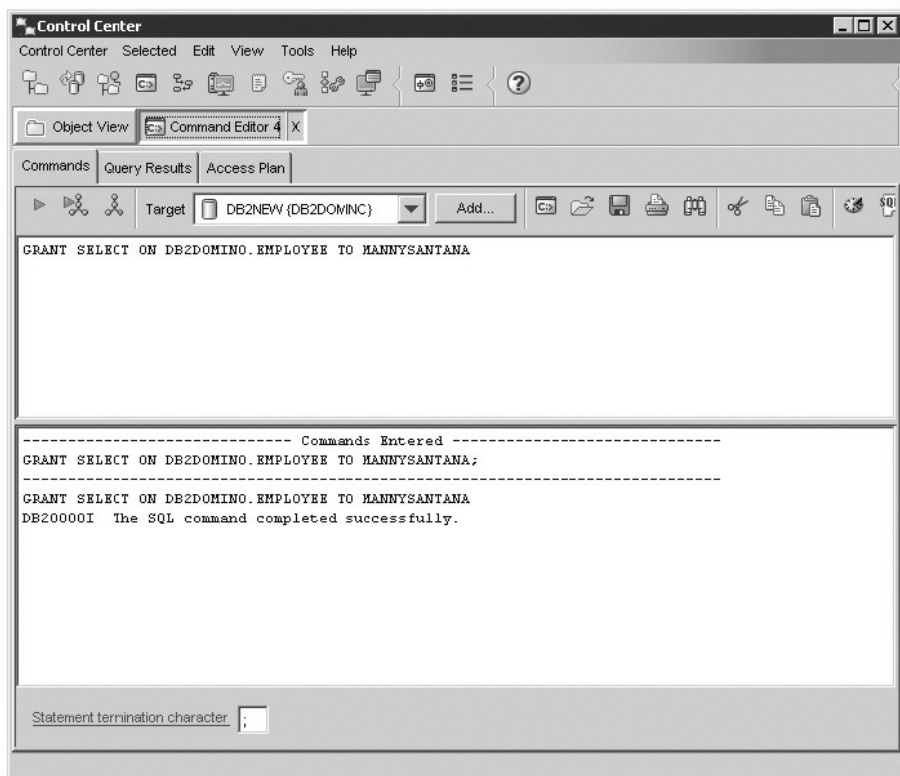


Рис. 2.29. DB2 Command Editor

Теперь вы можете создавать Query Views с интегрированными данными, благодаря чему ваше приложение Lotus Domino получает расширенные возможности доступа к данным.

2.4.2 Реализация доступа к интегрированным данным

DB2 в приложении ITSO Electronics

После завершения настройки интеграции на сервере DB2 мы можем получать доступ к интегрированным данным путем создания интегрированных Query Views в нашем приложении ITSO Electronics. Чтобы создать интегрированный Query View в приложении ITSO Electronics, выполните следующие действия.

1. Откройте базу данных ITSO Electronics Sales в Lotus Domino Designer и выберите пункт меню Create => View. Откроется окно Create View (Создать представление) (рис. 2.30). В поле "Имя" введите HR Employees ; под таким именем будет значиться ваш новый интегрированный Query View ; после чего установите переключатель Selection conditions в положение By SQL Query. Введите строку запроса SQL и щелкните по кнопке Save and Customize (Сохранить и настроить), чтобы настроить представление и добавить новые столбцы.

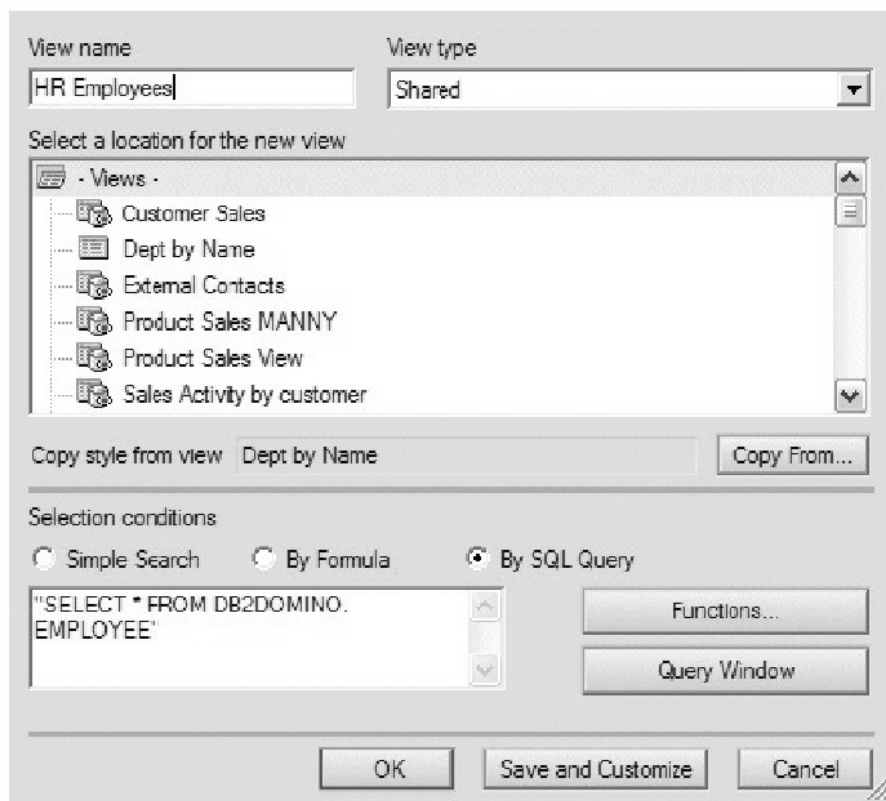


Рис. 2.30. Окно Create View

2. Добавьте новые столбцы, в которых будут отображены Employee No (порядковый номер), First name и Last Name из интегрированной таблицы Employee (рис. 2.31). Для каждого столбца определите соответствующее имя столбца из интегрированной таблицы в виде формулы значения столбца.

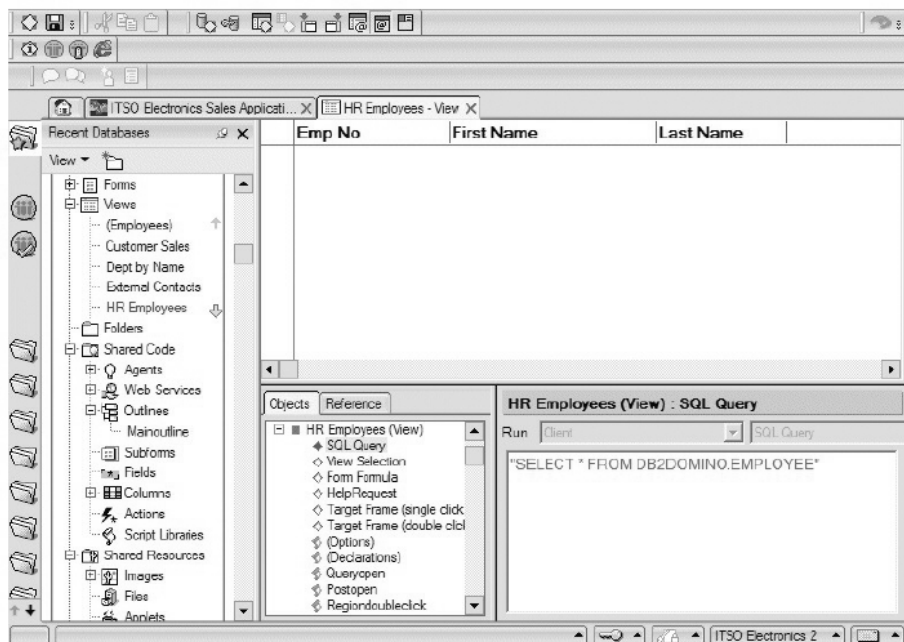


Рис. 2.31. Представление HR Employees в Domino Designer

- После сохранения представления и добавления представления в Outline выберите интегрированный Query View HR Employees в клиенте Lotus Notes, чтобы отобразить информацию из таблицы HR Employees, как показано на [рис. 2.32](#).

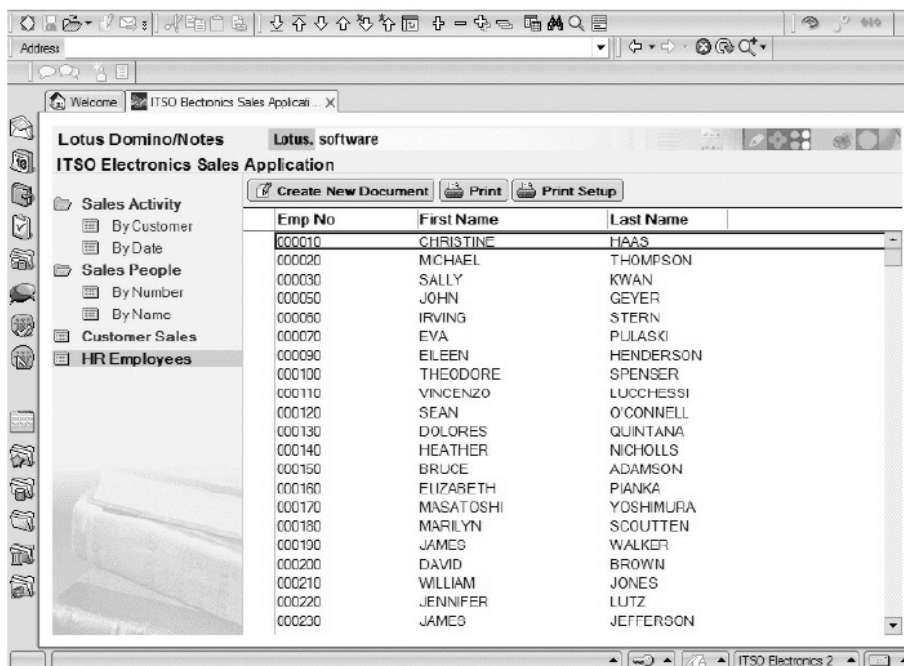


Рис. 2.32. Интегрированный Query View HR Employees

2.5 Вопросы при интеграции Lotus Domino и DB2

При интеграции Lotus Domino и DB2 следует иметь в виду, что:

- Хотя базы данных Lotus Domino хранятся на сервере DB2, списки контроля доступа (ACL) все равно продолжают поддерживаться, как и поля Readers и Authors.
- Когда вы сохраняете NSF в DB2, Lotus Domino создает и управляет набором таблиц, необходимых для поддержки ваших данных NSF, но вы не можете получить доступ к этим таблицам напрямую. Данные доступны лишь после определения DB2 Access View.
- Зашифрованные данные могут храниться в базе данных, основанной на DB2, но они недоступны в DB2 Access View.
- В Lotus Notes и Domino 7 при обращении к данным Notes на SQL через DB2 Access View защита на уровне пользователя всегда осуществляет проверку пользователя, обращающегося к этим данным, причем проверка осуществляется по связке имени

пользователя Notes и имени пользователя DB2.

- Если вы обращаетесь к DB2 Access View вне Notes/Domino на языке SQL, по умолчанию связка имени пользователя Notes и имени DB2 также используется.
- Если вы не хотите использовать привязку вне Notes, можно управлять доступом к DB2 Access View с помощью механизма DB2 GRANT, добавив приведенный ниже параметр к файлу NOTES.INI на сервере DB2 Access:

Allow_Anonymous_Access_From_DB2=значение

0 – отключает доступ по анонимной учетной записи;

1 – включает доступ по анонимной учетной записи через механизм DB2 GRANT.

Когда параметры файла NOTES.INI активированы, будет осуществляться анонимный доступ к Lotus Notes всякий раз, когда имя пользователя DB2 не связано с именем пользователя Lotus Notes.

Замечание Хотя активирование анонимного доступа и позволяет избавиться от привязки имен, Domino все же продолжает управлять доступом к DB2 Access View. Чтобы включить поддержку анонимного доступа, сервер Lotus Domino должен разрешить анонимный доступ и уровень доступа по умолчанию к базе данных (NSF), связанной с DB2 Access View.

- Если вы попытаетесь посмотреть на DB2 Access View в DB2 Control Center, а ваша учетная запись DB2 не позволяет получать доступ в DB2 Access View в Lotus Domino, то вы получите следующее сообщение об ошибке:

```
"com.ibm.db.DataException: A database manager error occurred.:  
[IBM][CLI Driver][DB2/NT]  
SQL0443N Routine "ISREADER3" (specific name "") has returned an err  
SQLSTATE with  
diagnostic text "14512 : No matching Notes user found for DB2 user and  
anony".  
SQLSTATE=04004
```

Дополнительную информацию по безопасности и интеграции DB2 можно получить в Lotus Domino 7 Designer Help в разделе DB2 Access View security по адресу: ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf

В этом разделе мы рассмотрели только те аспекты безопасности, которые связаны с интеграцией DB2 и Domino. Для получения более полного обзора по безопасности в Lotus Domino вы можете обратиться к готовящемуся к выходу руководству Security Considerations in Notes and Domino 7, REDP-4104.

2.6 Поиск и устранение неисправностей при интеграции DB2 и Lotus Domino

В этом разделе дается несколько советов по устранению проблем, связанных с интеграцией с DB2. Для получения дополнительных сведений перейдите к разделу DB2 Integration and Troubleshooting в справке Lotus Domino 7 Designer.

Если при интеграции Domino и DB2 возникают какие-либо проблемы, первое, что необходимо сделать, – проверить конфигурацию сервера DB2 Access. Инструмент DB2 Access Test тестирует все поля параметров сервера DB2 Access, начиная с серверного документа и заканчивая всеми параметрами сервера DB2 Access из файла NOTES.INI. Если все поля и параметры верны, этот инструмент приступает к проверке соединения между сервером DB2 Access и выбранным сервером Lotus Domino, проверяет, все ли функции и свойства присутствуют, определяет, корректен ли документ Connection на сервере DB2 Access, после чего пробует открыть директорию Domino на сервере DB2 Access. Если DB2 Access Test обнаруживает какие-либо проблемы, информация о неполадке возвращается в консоль сервера Lotus Domino или же в клиент Lotus Domino Administrator. Для того чтобы воспользоваться инструментом DB2 Access Test, сделайте следующее:

1. В Lotus Domino Administrator перейдите к закладке Configuration (Конфигурация).
2. Щелкните по кнопке Connections (Соединения). Выберите документ DB2 Access Server Connection.

3. В панели Tools (Сервис) выберите DB2 Server => Test DB2 Access (Сервер DB2 => Проверить доступ к DB2).

На рис. 2.33 показан результат работы инструмента DB2 Access Test.

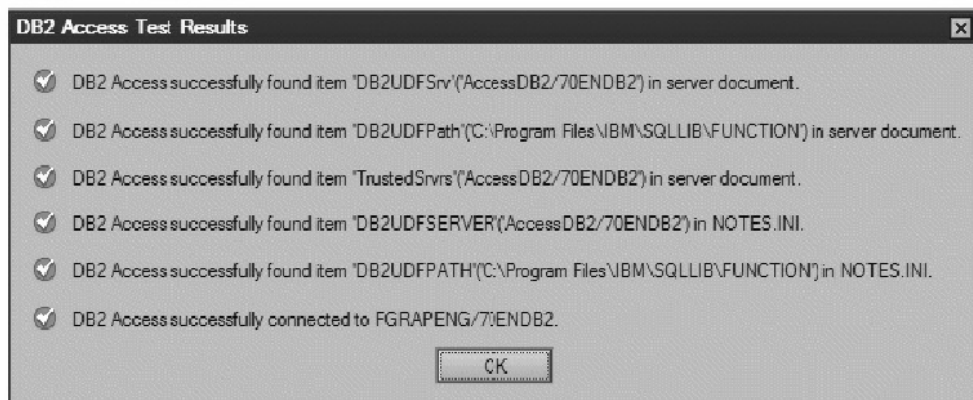


Рис. 2.33. Результат работы инструмента DB2 Access Test

2.6.1 Список контроля доступа (ACL), необходимый для DB2 Access Views

Если при использовании DB2 Access Views пользователи столкнутся с проблемой получения доступа к вашей базе данных Lotus Notes с поддержкой DB2, убедитесь в том, что элемент ACL для каждого пользователя идентичен первому элементу в поле User name в документе Person пользователя, хранящемся в директории Domino на сервере Domino.

2.6.2 Ошибки, возникающие при открытии Query View

Обычно, когда вы получаете сообщение об ошибке при открытии Query View, лучше всего взглянуть на серверную консоль на сервере Lotus Domino. Сообщения об ошибке, появляющиеся на серверной консоли, обычно напрямую указывают на источник той или иной проблемы.

В приведенном ниже списке представлены самые распространенные сообщения об ошибках и некоторая информация о них.

- DB2-Authorization Violation (Нарушение авторизации DB2).

Хотя сообщение об ошибке выглядит именно так, данная проблема никак не связана с безопасностью. Такую информацию можно увидеть на серверной консоли:

- Failed when DB2 was collecting column info. DB2-Authorization violation.: 42S02 - [IBM][CLI Driver][DB2/NT] SQL0204N "SALES.SALESDAV" is an undefined name. SQLSTATE=42704 - *LOCAL.DB2.051103024144 (Ошибка DB2 при сборе информации о столбцах. Нарушение авторизации DB2.: 42S02 - [IBM][CLI Driver][DB2/NT] SQL0204N "SALES.SALESDAV" не определенное имя. SQLSTATE=42704 - *LOCAL.DB2.051103024144).

Существует три возможные причины появления этого сообщения об ошибке:

- схема DB2 Sales в данном примере не существует;
- доступ DB2 SALESDAV в этом примере не существует;
- DB2 Access View не заполнен.
- Failed when DB2 was collecting column info. DB2-Authorization violation.: 42S22 - [IBM][CLI Driver][DB2/NT] SQL0206N "C.CUSTOMERNAM" is not valid in the context where it is used. SQLSTATE=42703 - *LOCAL.DB2.051103024144 (Ошибка DB2 при сборе информации о столбцах. Нарушение авторизации DB2.: 42S22 - [IBM][CLI Driver][DB2/NT] SQL0206N "C.CUSTOMERNAM" не верен в контексте, в котором он использован SQLSTATE=42703 - *LOCAL.DB2.051103024144).

Одной из причин появления такой ошибки является то, что столбец, к которому был применен запрос, не существует в DB2 Access View; в этом примере таким столбцом является CUSTOMERNAM.

- DB2-Dynamic SQL Error (Ошибка SQL в динамической DB2).

Это сообщение об ошибке указывает на синтаксическую ошибку в SQL-выражении. Посмотрите на серверную консоль и проверьте SQL-выражение:

```
Failed when DB2 was collecting column info.  
DB2-Dynamic SQL error.: 42601 - [IBM][CLI  
Driver][DB2/NT] SQL0104N An unexpected token  
"FROMSALES2" was found following "SELECT *".  
Expected tokens may include: "<from>".  
SQLSTATE=42601 - G921557D. GD04.0088C6170623  
(Ошибка DB2 при сборе информации о столбцах. Ошибка SQL в  
динамической DB2.: 42601 - [IBM][CLI Driver]  
[DB2/NT] SQL0104N неожиданный символ "FROMSALES2"  
нашел следующее: SELECT *. Expected tokens may  
include: "<from>". SQLSTATE=42601 -  
G921557D.GD04.0088C6170623 ).
```

В этом примере в команде было пропущено место между FROM и SALES2.

- Opening View: The given query cannot generate a result set and therefore has not been executed (Открытие представления: сделанный запрос не может сгенерировать набор результатов и, следовательно, не может быть выполнен).

Из соображений безопасности SQL-выражения в Query View были ограничены до выражений, которые возвращают наборы результатов. Команды Update и Insert не работают, когда используются для выборки. Например, если вы хотите выполнить в качестве первой команды SELECT и допустили ошибку при

наборе SQL-выражения, такую как написание команды `SELECT` вместо `SELECT`, это может вызвать ошибку.

- This function is not implemented on this version of the server (Эта функция неприменима для этой версии сервера).

Вероятнее всего, вы пытаетесь открыть Query View на локальной реплике или на системе, в которой не включена поддержка DB2.

- Failed when DB2 was fetching a note. DB2-External function exception.: 38503 - [IBM] [CLI Driver][DB2/NT] SQL0430N User defined function "DOMINO.ISREADER3" (specific name "SQL051024022301200") has abnormally terminated. SQLSTATE = 38503 - *LOCAL.DB2.051028175704 (Ошибка DB2 при выборке записи. DB2-внешнее исключение функции.: 38503 - [IBM] [CLI Driver][DB2/NT] SQL0430N. Функция "DOMINO.ISREADER3" (собственное имя "SQL051024022301200"), определенная пользователем, была аварийно завершена. SQLSTATE=38503 - *LOCAL.DB2.051028175704).

Одна из возможных причин появления данной ошибки заключается в существовании более ранней версии Lotus Notes и Domino на данной системе. Для решения данной проблемы достаточно будет просто удалить более раннюю версию или переименовать папку.

Совет Упростите код или конструкцию вашего приложения пошаговым образом до тех пор, пока ошибки не прекратят появляться. Очень часто после выполнения такой операции раскрываются истинные причины той или иной проблемы. Во многих случаях решение проблемы становится очевидным.

2.6.3 Данные, которые могут оказаться полезными при

ПОИСКЕ ИСТИННЫХ ИСТОЧНИКОВ ПРОБЛЕМ

Часто оказывается полезным собрать информацию, касающуюся диагностики. Это поможет вам, вашим коллегам или персоналу IBM найти источники проблемы.

- Точная информация о программных компонентах, которые вы используете:
 - операционная система;
 - сервер DB2;
 - клиент Runtime;
 - Lotus Notes и Domino;
 - Сервер DB2 Access.
- Информация об используемых патчах и локализациях.
- Снимки экранов со всеми сообщениями об ошибках.
- В случае нарушения работы одного из перечисленных ниже компонентов будет создан файл NSD с информацией о диагностике:
 - клиент Lotus Notes;
 - сервер Lotus Domino;
 - сервер DB2 Access.
- Вся доступная информация из журнала.
- Дизайн базы данных.
- Пошаговое описание действий, совершенных до того, как появилась проблема.
- Файл NOTES.INI с сервера Lotus Domino и с сервера DB2 Access.

Улучшение работы приложения фирмы ITSO Electronics с помощью Web-сервисов

Лекция посвящена вопросам создания и использования Web-сервисов в Lotus Domino 7 на примере ITSO Electronics

С появлением поддержки Web-сервисов Lotus Domino Designer 7 продолжает традицию поддержки новых стандартов для расширения возможностей его модели быстрой разработки приложений. Web-сервисы позволяют превратить приложения Lotus Domino в Web-сервисы, базирующиеся на сервере Lotus Domino, таким образом увеличивая возможность доступа к данным и обеспечивая обращение к приложениям через множество различных платформ. Благодаря тому, что все они построены на основе открытых стандартов, Web-сервисы – автономные, модульные приложения, способные работать друг с другом, без необходимости полагаться на соединения, запрограммированные пользователем. Так как Web-сервисы позволяют различным приложениям и информационным системам взаимодействовать друг с другом, серверные компоненты могут быть скомбинированы и рекомбинированы на лету, что позволяет компаниям быстро реагировать на постоянно меняющиеся потребности бизнеса и клиентов.

В этой лекции мы проанализируем, как улучшается приложение ITSO Electronics благодаря использованию Web-сервисов. Для этого рассмотрим следующие темы:

- сервис-ориентированная архитектура: как Web-сервисы согласовываются с SOA (Server-Oriented Architecture);
- Web-сервисы и язык описания Web-сервисов (Web Services Description Language, WSDL);
- коммерческие предпосылки использования Web-сервисов;
- элемент Web-сервиса: добавление простых и комплексных элементов Web-сервиса к приложению ITSO Electronics;
- экспорт и импорт WSDL-документа;
- обработка ошибок и исключений в Web-сервисах;
- безопасность Web-сервисов;
- использование Web-сервисов на языках программирования Java и LS2J/Lotus-Script.

3.1 Сервис-ориентированная архитектура и Web-сервисы

Сервис-ориентированная архитектура (SOA) является компонентной моделью, позволяющей взаимодействовать друг с другом различным функциональным модулям приложения, называемым сервисами, через строго определенный интерфейс (*well-defined interface*) и набор условий между этими сервисами. Интерфейс определен независимым способом, т. е. он должен быть независим от аппаратной платформы, операционной системы и языка программирования, на котором этот сервис реализован. Таким образом, становится возможным взаимодействие друг с другом сервисов, построенных на различных системах унифицированным и универсальным способом. Так как Web-сервисы характеризуются независимым от платформы подходом к разработке приложения, они являются неотъемлемым компонентом архитектуры SOA-приложения.

Эта особенность, заключающаяся в независимом описании интерфейса, который не строго привязан к конкретной реализации, известна как слабая связь (*loose coupling*) между сервисами. Преимущество системы со слабой связью заключается в ее гибкости и способности сохранять работоспособность после эволюционных изменений в структуре и реализации функций каждого сервиса и всей программы в целом. Тем не менее, тесно связанные системы предполагают, что интерфейсы между различными компонентами приложения тесно взаимосвязаны, что делает их более непрочными, когда необходимо произвести изменения частей или всего приложения.

Необходимость в системах со слабой связью растет из-за того, что с каждым днем возрастает потребность в более гибких бизнес-приложениях, которая, в свою очередь, основывается на том, что бизнес должен адаптироваться к изменчивости внешней среды, например к постоянно меняющимся принципам стратегии, устойчивости коммерческого положения, бизнес-целям, партнерству, состоянию промышленности, а также другим факторам, связанным с бизнесом, которые оказывают влияние на природу бизнеса. При желании вы можете обратиться к бизнесу, быстро реагирующему на изменения окружающей среды, т. е. к бизнесу по требованию (*On Demand Business*),

который меняется в зависимости от того, как реализуются цели.

Сервис-ориентированная архитектура не только является новшеством, но и представляет собой альтернативную модель более традиционным, жестко связанным (*tightly coupled*) объектно-ориентированным моделям, появившимся в последние десятилетия. Тогда как системы, базирующиеся на SOA, не исключают того факта, что отдельные сервисы сами могут быть построены по объектно-ориентированной схеме, схема всей системы сервис-ориентированна. Поскольку данное правило распространяется на объекты, находящиеся в пределах системы, SOA основана на объектах, но в целом не объектно-ориентированная. Различие заключается в самих интерфейсах. Классическим примером одной из самых первых SOA-систем является *Common Object Request Broker Architecture (CORBA)*, оперирующая принципами, схожими с SOA.

В настоящее время SOA использует преимущества языка XML (*Extensible Markup Language*). В результате описания интерфейсов на языках, основанных на XML, называемых языками описания Web-сервисов (*Web Services Definition Language, WSDL*)), Web-сервисы обрели более гибкие и динамичные системные интерфейсы по сравнению с разработанным ранее языком описания интерфейса (*Interface Definition Language, IDL*), использованным в CORBA.

Помимо Web-сервисов существуют и другие способы реализации SOA, такие как CORBA и межплатформенные системы, ориентированные на информационный обмен, например IBM WebSphere® MQ. Однако для того чтобы превратить ее в архитектурную модель, вам понадобится не просто описание сервиса. Необходимо описать то, как все приложение будет осуществлять последовательность операций между сервисами. Кроме того, необходимо найти подходящее для вашего бизнеса программное обеспечение. Следовательно, SOA должна быть связана с коммерческими бизнес-процессами подобно тому, как ITSO Electronics ориентируется на технические процессы и отображает взаимосвязь между технической и коммерческой стороной дела. Например, плата поставщику за услуги – бизнес-процесс, в то время как обновление базы данных ваших электронных компонентов с целью внесения новых компонентов – процесс технический. Таким образом, технологический процесс играет значительную роль в схеме SOA.

3.2 Web-сервисы и WSDL

Web-сервис представляет собой программную систему, предназначенную для поддержки взаимодействия компьютеров через сеть. Интерфейс Web-сервиса описан в формате, который компьютер способен обработать (конкретно WSDL). Другие системы взаимодействуют с Web-сервисом способами, предусмотренными его описанием, с использованием сообщений SOAP, которые обычно передаются посредством HTTP с XML-преобразованием вместе с другими стандартами, связанными с Web. Lotus Domino Designer 7 включает элемент дизайна "Web-сервис", а также встроенную поддержку языка описания Web-сервисов (WSDL). WSDL – это стандарт спецификации для описания сервисов, работающих по сети и основанных на XML. Он обеспечивает поставщиков услуг простым способом описания основного формата запросов к системам вне зависимости от того, откуда будут вызываться сервисы.

Ограничения в реализации Web-сервисов в Lotus Domino 7 выражаются в следующем:

- На "родном" уровне поддерживаются только те элементы, которые предоставляются провайдером.
- Связь должна выполняться по простому протоколу доступа к объектам (SOAP) с помощью протоколов HTTP POST.

Элемент дизайна "Web-сервис" характеризуется следующими параметрами:

- Скомпилированный Web-сервис, подобно Web-агенту, представляет собой отдельную автономную программу в базе данных Lotus Domino. Для того чтобы им воспользоваться, необходимо включить Web-сервис или отреплицировать его на сервер, к которому имеется Web-доступ. К нему можно получить доступ с помощью одной из приведенных ниже URL-команд Lotus Domino:
 - ?OpenWebService: используя протокол HTTP POST, запускает Web-сервис. Если использован протокол HTTP GET, то команда ?OpenWebService вернет некоторую

информацию об этом сервисе.

- ?WSDL: запрашивает у Web-сервиса его WSDL-документ.
- Web-сервис можно протестировать во время HTTP-сессии в режиме предварительного просмотра Lotus Notes или Lotus Domino Designer.
- Web-сервис имеет те же возможности по обеспечению безопасности, что и агент.
- Вы можете импортировать существующий WSDL-документ, чтобы сгенерировать каркас Web-сервиса. Каркасный код соответствует описанию Web-сервиса. Затем вы добавляете исполнительный код.
- В качестве альтернативы можно приступить к созданию Web-сервиса с нуля, написав собственный код на языке Java или LotusScript, чтобы создать описание Web-сервиса.

3.2.1 Документ WSDL

Документ WSDL определяет Web-сервис и включает в себя приведенные ниже элементы:

- Сервис (*Service*).

Web-сервис в целом. В документе WSDL сервис описывается элементом `<definitions>` в корне дерева. Дочерними являются элементы `<types>`, `<message>`, `<port-Type>`, `<binding>`, `<service>`. У элемента `<service>` есть атрибут `"name"`, который используется для того, чтобы присвоить имя сервису.

- Типы (*Types*).

Типы данных, используемые сервисом. В документе WSDL типы данных описываются с помощью элемента `<types>`, содержащего один или более элементов `<xsd:schema>`. Для получения дополнительной информации, касающейся описания типов данных, обратитесь по адресу: ссылка: <http://www.w3.org/TR/xmlschema-2>

- Сообщение (Message).

Абстрактное описание данных, передаваемых или принимаемых сервисом. Допустимо любое количество сообщений. В документе WSDL каждое сообщение описывается элементом `<message>`. Каждый элемент `<message>` содержит один или более элементов `<part>`, необходимых для описания данных по имени и типу.

- Тип порта (PortType).

Абстрактный набор операций, поддерживаемых сервисом. В документе WSDL тип порта описывается элементом `<portType>`. Каждый элемент `<portType>` содержит один или более элементов `<operation>`.

- Операция (Operation).

Абстрактное описание действия, поддерживаемого сервисом. В документе WSDL операция описывается элементом `<operation>`. Каждый элемент `<operation>` содержит элементы `<input>` и `<output>`, определяющие связанные сообщения. Дополнительные элементы `<fault>` также могут быть определены.

- Привязка (Binding).

Определение протокола и формата данных, предназначенных для типа порта. В документе WSDL привязка описывается элементом `<binding>`. У элемента `<binding>` есть атрибут `"type"`, который определяет название типа порта. Для кодирования SOAP используется элемент `<binding>`, содержащий элемент `<soap:binding>` с атрибутами `"style"` (`"rpc"` или `"document"`) и `"transport"` ("ссылка: <http://schemas.xmlsoap.org/soap/http>"). Элемент `<binding>` содержит элементы `<operation>`, описывающие формат данных для каждой операции.

- Порт (Port).

Конечная точка, определенная в виде комбинации связывающего и сетевого адресов. В документе WSDL порт описывается элементом `<port>` под элементом `<service>`. У элемента `<port>` есть атрибут `"binding"`, необходимый для присвоения имени связыванию, и атрибут `"address location"`, необходимый для присвоения имени конечной точке сети, связанной с привязкой.

3.2.2 Простой протокол доступа к объектам (SOAP)

Элемент Web-сервисов Lotus Domino 7 обеспечивает средой для хостинга, основанной на простом протоколе доступа к объектам (SOAP) версии 1.1. SOAP определяет информацию, базирующуюся на XML и используемую для обмена структурированной и классифицированной информацией между равноправными участниками сети в децентрализованной, распределенной среде. Сообщение SOAP, по существу, представляет собой одностороннюю передачу от отправителя к получателю SOAP, однако предполагается, что сообщения SOAP комбинируются с помощью приложений для применения более сложных схем взаимодействия.

Из положения консорциума Всемирной паутины (W3C):

"SOAP is a *lightweight protocol* for exchanging structured information in a *decentralized* distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses¹⁾".

("SOAP представляет собой облегченный протокол, предназначенный для обмена структурированной информацией в децентрализованной и распределенной среде. Он основан на протоколе XML, состоящем из трех частей: оболочки, определяющей структуру для описания сообщения и процесса его обработки, набора правил кодирования, служащего для выражения экземпляров типов данных, определенных приложением, и правила, предназначенного для представления удаленных процедурных запросов и ответов".)

3.2.3 Импортированный WSDL

В Lotus Domino Designer 7 разработчик приложения может импортировать существующий файл WSDL в приложение и Domino Designer, затем сгенерировать классы на языке LotusScript или Java. Тем не менее, разработчики должны иметь в виду, что не все существующие WSDL-файлы могут быть интерпретированы Lotus Domino Designer 7. Некоторые файлы могут быть отклонены, в зависимости от используемых элементов XML.

3.3 Коммерческие предпосылки использования Web-сервисов

Начиная с Domino Release 5 появилась возможность написания приложений вне сервера Lotus Domino и получения доступа к данным Lotus Domino напрямую. Этого удалось достичь за счет использования привязок COM и CORBA к внутренним классам. Тем не менее, проблема проверки достоверности и согласованности данных, возможно, не была решена соответствующим образом с помощью внешних приложений. Web-сервисы предоставляют хороший компромисс между прямым доступом к данным и возможностью выполнения сетевых операций. Они предоставляют способ раскрытия функций приложений и данных в сети. Разработчики приложений на языке Java, Microsoft .NET, а также многих других могут получить доступ к данным Lotus Domino, просто используя Web-сервисы.

Преимущества приложения, в котором внедрены Web-сервисы, лучше всего продемонстрировать на приложении ITSO Electronics, которое предоставляет клиентам текущую информацию о продуктах. Например, если приложение ITSO Electronics обещает пользователям доступ к информации о продуктах компании ITSO Electronics, ему необходимо разрешить доступ внешних приложений к базе данных покупателей вне зависимости от используемого приложения, будь то Java, J2EE или .NET. В связи с тем, что приложение покупателя отправляет запрос о получении информации Web-сервису, оно не должно зависеть от платформы и операционной системы. Web-сервис получает запрос о получении данных в виде сообщения XML. Следовательно, если

пользователь введет запрос в форму приложения покупателя с целью получения информации о продукте компании ITSO Electronics, Web-сервис получит запрос в виде сообщения SOAP в формате XML и отправит ответ клиенту таким же образом. Используя расширенную поддержку, приложение ITSO Electronics может расширить свою функциональность в целях получения информации о цене продуктов от множества производителей. Используя Java, Web-сервис может отправлять запросы на получение данных на множество URL-адресов, запрашивая данные о цене, используя протокол XML. Все запросы отправляются в виде сообщений SOAP, и ответы приходят в таком же формате, в приложении ITSO Electronics. Пользователь при этом не знает ни о типе базы данных, из которой получает данные, ни об операционной системе, ни о платформе. Пользователи получают доступ, по-видимому, к неограниченному запасу данных, представленному с большой эффективностью.

3.4 Добавление Web-сервисов к приложению ITSO Electronics

Используя Web-сервисы и поддержку WSDL, вы можете быстро создавать или расширять код Java или LotusScript, открывая, таким образом доступ к тысячам приложений Lotus Domino из внешних систем, таких как Java 2 Platform, Enterprise Edition (J2EE) и приложениям Microsoft .NET. В этом разделе мы показываем, как улучшаются приложения ITSO Electronics путем внедрения Web-сервисов в среду разработки приложений Lotus Domino.

Замечание Web-сервисы могут быть добавлены к любой базе данных Lotus Notes и Domino. Тем не менее, для достижения цели, поставленной в этой книге, все Web-сервисы добавлены к двум отдельным базам данных в приложении ITSO Electronics, которые называются `webservices.nsf` и `wsconsumer.nsf`.

3.4.1 Коммерческие предпосылки добавления Web-сервисов к приложению ITSO Electronics

Основным нашим требованием по внедрению Web-сервисов в

приложение ITSO Electronics стала необходимость в просмотре покупателям и текущей информации о наших продуктах. Web-сервисы лучше всех справятся с подобной задачей. В будущем компания ITSO Electronics планирует использовать Web-сервисы для того, чтобы позволить покупателям получать доступ к информации о счетах, вне зависимости от используемой платформы, операционной системы или интерфейса.

3.4.2 Пользовательские требования

Основным пользователем Web-сервисов приложения ITSO Electronics в нашем примере будет клиент фирмы ITSO Electronics. Приложение необходимо для обеспечения доступа пользователя к текущей информации о продукте через программный механизм и пользовательский интерфейс, интуитивный и легкий в использовании.

3.5 Создание простого Web-сервиса в Lotus Domino Designer 7

Позвольте продемонстрировать вам, насколько легко создать Web-сервис с помощью IBM Lotus Domino Designer 7.

Начнем с исключения ошибок, обработки особых ситуаций, регистрации и производительности. Сосредоточимся на функциональности ядра.

Необходимо, чтобы приложение ITSO Electronics поддерживало и Java, и Lotus-Script.

3.5.1 Создание нового Web-сервиса

В этом разделе иллюстрируется создание нового Web-сервиса. В качестве примера используется приложение ITSO Electronics:

1. Откройте базу данных в Domino Designer. В нашем примере мы выбрали базу данных ITSO Web Services. Из списка дизайна

выберите Shared Code => Web Services (Общий код => Web-сервисы), затем щелкните по кнопке New Web Service (Создать Web-сервис). В окне свойств Web-service в поле Name (Название) введите ProductInfo, таким образом вы присвоите имя Web-сервису (см. [рис. 3.1](#)).

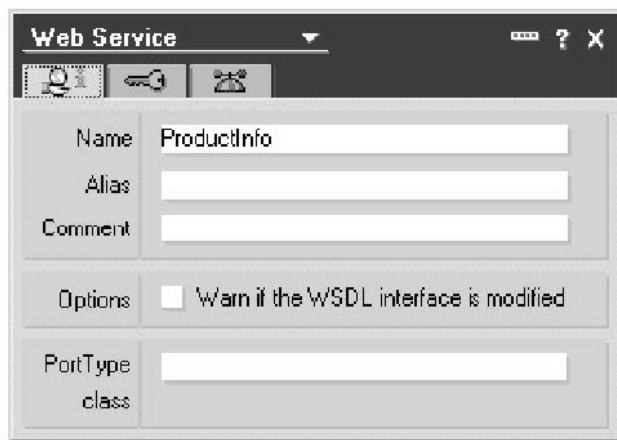


Рис. 3.1. Окно свойств Web Service

- Чтобы создать описание LotusScript, как на [рис. 3.2](#), вставьте или наберите код, приведенный в [примере 3.1](#), в раздел Web-сервиса Declaration.

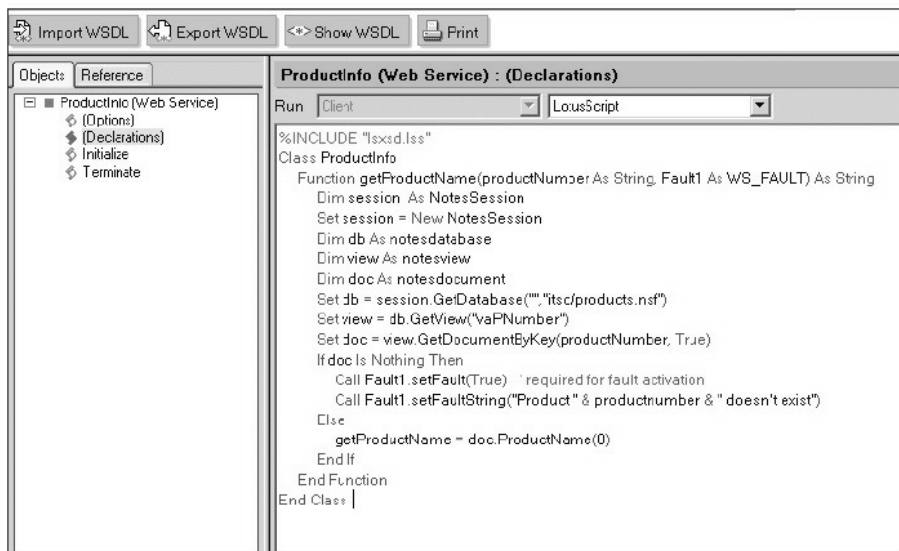


Рис. 3.2. Исходный код для простого Web-сервиса

```

%INCLUDE "lsxsd.lss"
Class ProductInfo
Function getProductName(productNumber As String, Fault1 As WS_FAULT1) As String
Dim session As NotesSession
Set session = New NotesSession
Dim db As notesdatabase
Dim view As notesview
Dim doc As notesdocument
Set db = session.GetDatabase("", "itso/products.nsf")
Set view = db.GetView("vaPNumber")
Set doc = view.GetDocumentByKey(productNumber, True)
If doc Is Nothing Then
Call Fault1.setFault(True) ' необходимо для ошибочной активации
Call Fault1.setFaultString("Product " & productnumber & " doesn't exist")
Else
getProductName = doc.ProductName(0)
End If
End Function
End Class

```

Пример 3.1. Код для простого Web-сервиса

3. Снова откройте окно свойств Web Service и в поле Port type class введите ProductInfo, перед тем как сохранять Web-сервис, как показано на [рис. 3.3](#).



Рис. 3.3. Окно свойств Web Service: поле PortType class

4. Перейдите к закладке *Advanced* (Дополнительно) и убедитесь в том, что значение по умолчанию для переключателя *Programming model* (Программируемая модель) установлено в положение *RPC* и значение поля *SOAP message format* (формат сообщения SOAP) установлено на *RPC/encoded* (см. рис. 3.4).

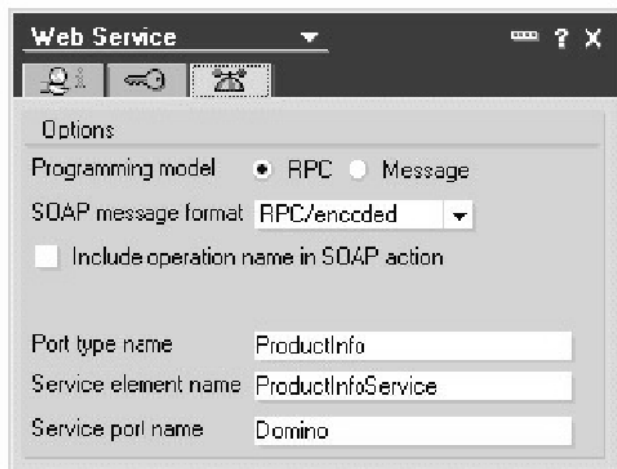


Рис. 3.4. Окно свойств Web Service: закладка *Advanced*

5. Закройте окно свойств и сохраните Web-сервис.

3.5.2 Просмотр Web-сервиса через браузер

Для того чтобы протестировать новый Web-сервис *ProductInfo*, откройте браузер и введите следующий URL:

ссылка: <http://servername/databasename.nsf/webservicename?openwebservice>
(ссылка: <http://имясервера/названиебазыданных.nsf/названиеWeb-сервиса?openwebservice>)

В нашем сценарии для приложения *ITSO Electronics* мы использовали приведенный ниже URL:

ссылка:

<http://domino7appdev.cam.itso.ibm.com/itso/webservices.nsf/ProductInfo?openwebservice>

Lotus Domino ответит на этот запрос именем порта Web-сервиса и названиями операций для этого порта, а также выведет ссылку на документ WSDL, как показано на рис. 3.5.



Рис. 3.5. Web-сервис, открытый в Web-браузере

3.5.3 Сгенерированный документ WSDL

Приведенная ниже URL-команда Lotus Domino возвращает документ WSDL для Web-сервиса в ответ на HTTP GET, используя суффикс ? WSDL:

ссылка: <http://servername/databasename.nsf/webservicename?WSDL> (ссылка: <http://названиесервера/названиебазыданных.nsf/названиеWeb-сервиса?WSDL>)

Например, в нашем сценарии была использована следующая команда:

ссылка:

<http://domino7appdev.cam.itso.ibm.com/itso/webservices.nsf/ProductInfo?WSDL>

В примере 3.2 показан полученный в результате WSDL-документ.

Для более подробного описания документа WSDL см. п. 3.2.1, "Документ WSDL".

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

<wsdl:definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:
apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:DefaultNamespace" xmlns:intf="urn:DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://
schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www
w3.org/2001/XMLSchema"
targetNamespace="urn:DefaultNamespace">
  <wsdl:message name="GETPRODUCTNAMERequest">
    <wsdl:part name="PRODUCTNUMBER" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="GETPRODUCTNAMEResponse">
    <wsdl:part name="GETPRODUCTNAMEReturn" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="ProductInfo">
    <wsdl:operation name="GETPRODUCTNAME" parameterOrder="PRODU
    <wsdl:input message="impl:GETPRODUCTNAMERequest"
    name="GETPRODUCTNAMERequest"/>
    <wsdl:output message="impl:GETPRODUCTNAMEResponse"
    name="GETPRODUCTNAMEResponse"/>
  </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="DominoSoapBinding" type="impl:ProductInfo">
    <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/htt
    <wsdl:operation name="GETPRODUCTNAME">
      <wsdlsoap:operation soapAction=""/>
      <wsdl:input name="GETPRODUCTNAMERequest">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:DefaultNamespace"
use="encoded"/>
      </wsdl:input>
      <wsdl:output name="GETPRODUCTNAMEResponse">

<wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:DefaultNamespace"
use="encoded"/>
      </wsdl:output>
    </wsdl:operation>

```

```
</wsdl:binding>
<wsdl:service name="ProductInfoService">
  <wsdl:port binding="impl:DominoSoapBinding" name="Domino">
    <wsdlsoap:address
location="http://domino7appdev.cam.itso.ibm.com:80/itso/webservices.nsf/
ProductInfo?OpenWebService"/>
  </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

Пример 3.2. Код, полученный от сервера при обращении к Web-сервису с суффиксом ?WSDL

Полученный в результате генерации с суффиксом ?WSDL код сложен, однако это не важно, так как документы WSDL считываются компьютерами, а не пользователями. Например, Lotus Domino Designer может импортировать документ WSDL и создавать каркасный класс в LotusScript или Java. Обсуждением данной темы мы занимались в [п. 3.2.3](#), "Импортированный WSDL".

3.5.4 Тестирование простого Web-сервиса

Существует множество приложений, которые позволяют тестировать и использовать Web-сервисы. Для достижения целей, поставленных в этой книге, мы будем использовать IBM Rational [®] Application Developer. Чтобы начать тестирование Web-сервисов, используя Rational Application Developer, выполните приведенную ниже последовательность действий.

1. Запустите Rational Application Developer.
2. Выберите Run => Launch the Web Services Explorer (Выполнить => Запустить обозреватель Web-сервисов).
3. Щелкните по иконке WSDL page (вторая справа, находится в верхней правой части основного окна).
4. Затем перейдите к панели Navigator (Навигатор) и выберите WSDL Main, как показано на [рис. 3.6](#).

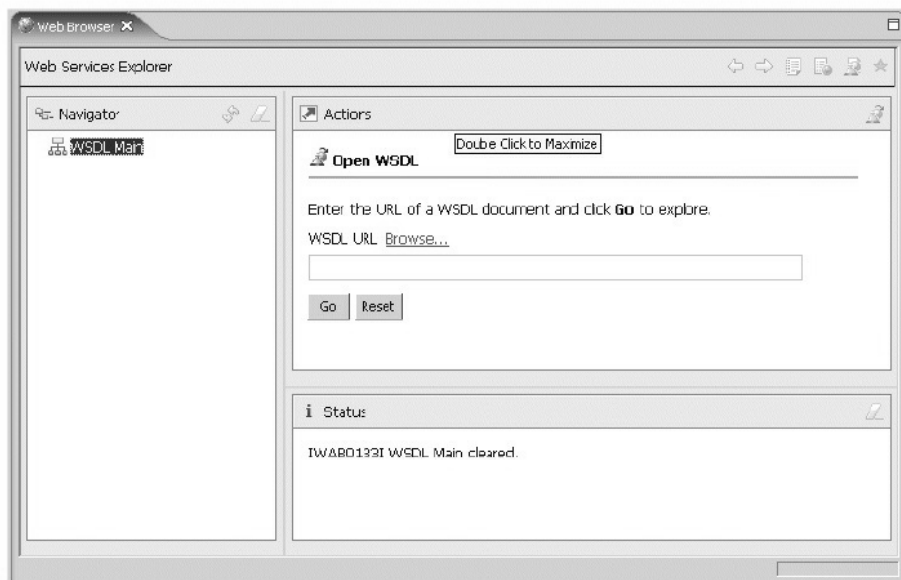


Рис. 3.6. Выберите WSDL Main на панели Navigator

5. Перейдите к панели Open WSDL, после чего определите WSDL URL. В нашем приложении ITSO Electronics мы использовали следующий URL:

ссылка:

<http://domino7appdev.cam.itso.ibm.com/itso/webservices.nsf/ProductInfo?wsdl>

6. Щелкните по кнопке Go (Перейти), после чего откроется панель WSDL Binding Details (Детали привязки WSDL), которая представлена на рис. 3.7.

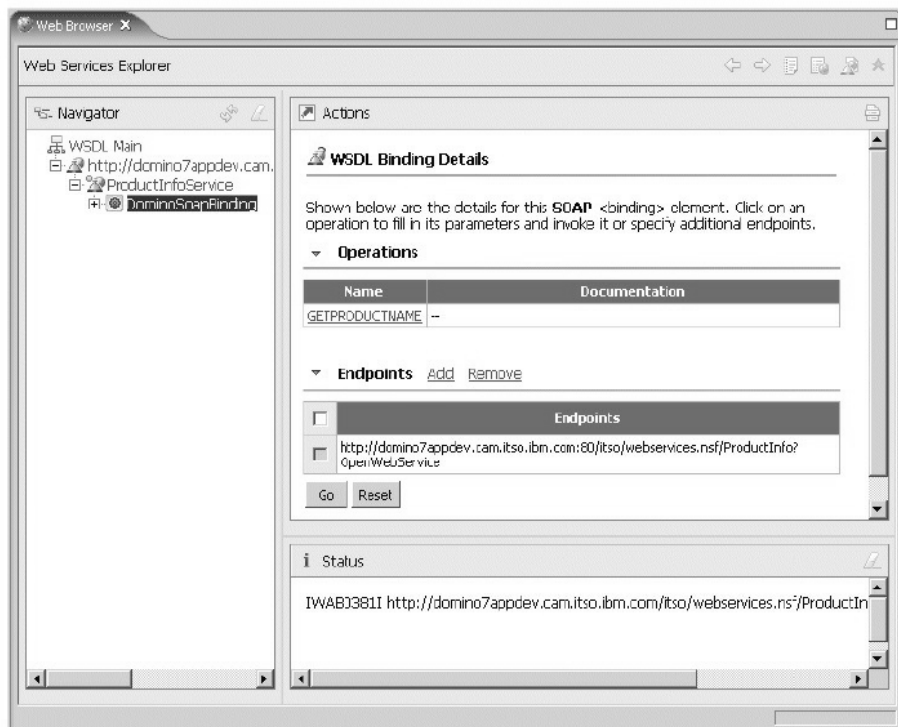


Рис. 3.7. Панель WSDL Binding Details

7. В панели WSDL Binding Details показаны операции, производимые для элемента SOAP. Выберите операцию, указанную в панели, чтобы записать ее параметры и задействовать ее или же определить большее количество конечных точек. В этом сценарии (сценарии компании ITSO Electronics) выберите операцию GETPRODUCTNAME (Получить название продукта), после чего появится панель Invoke a WSDL Operation (Задействовать операцию WSDL), как показано на рис. 3.8.

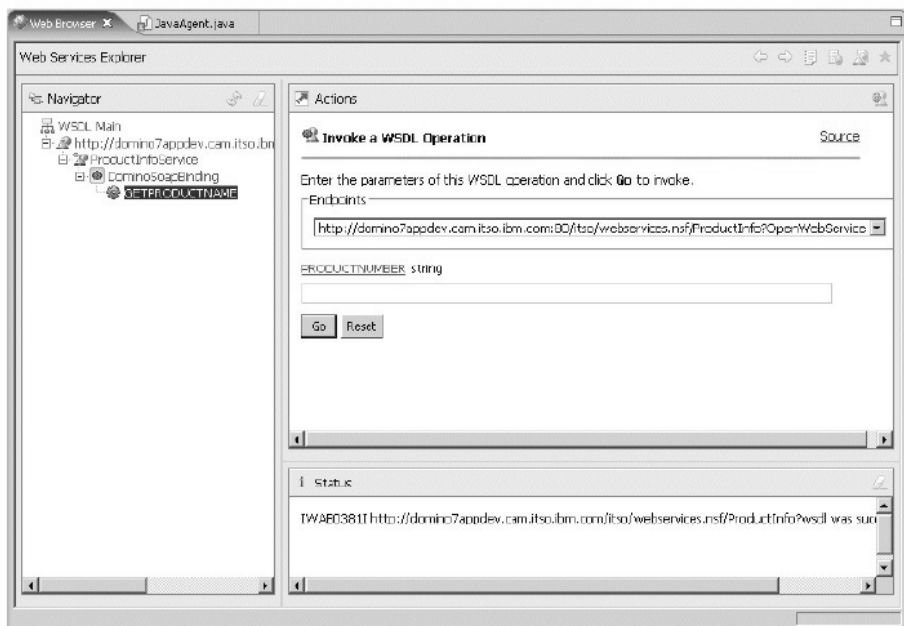


Рис. 3.8. Панель Invoke a WSDL Operation

8. Введем значение `PRODUCTNUMBER` и щелкнем по кнопке `Go`. Результат появился в панели `Status` (Статус). В нашем сценарии в качестве `PRODUCTNUMBER` (номера продукта) мы ввели значение `45000` и получили результирующую строчку `Redbook Basic` (см. [рис. 3.9](#)).

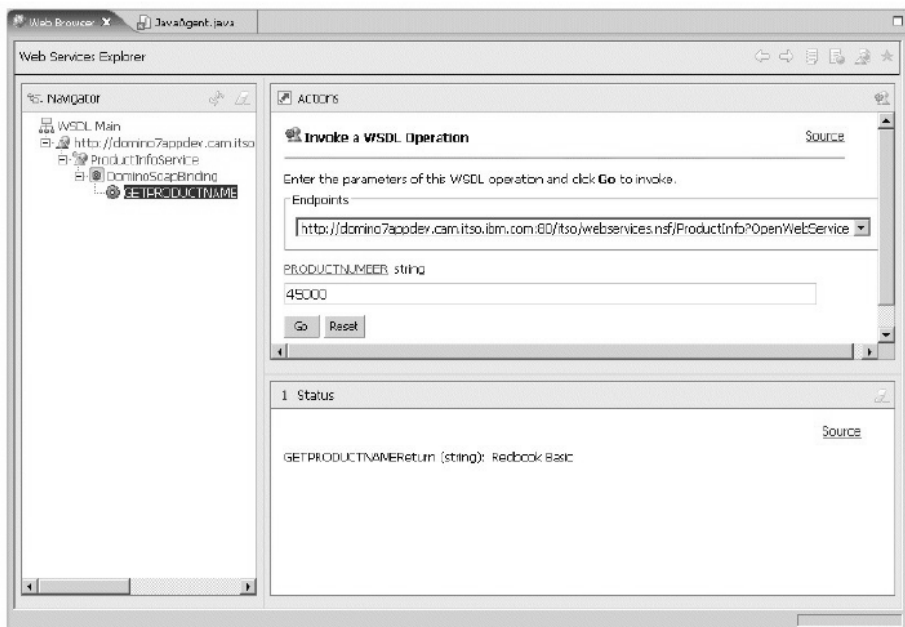


Рис. 3.9. Панель Invoke a WSDL Operation Status

9. Чтобы увидеть, что на самом деле происходит "за кулисами", щелкните по кнопке *Source* (Источник), находящейся на панели *Status*. В примере 3.3 приведен код, отправленный на сервер Lotus Domino в контексте запроса SOAP. Код, которым сервер Lotus Domino ответил отправлением контекста ответа SOAP на этот запрос, представлен в примере 3.4, после чего этот код был представлен в панели *Status*, как и видно из рис. 3.10.

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <SOAP-ENV:Body>
- <ns0:GETPRODUCTNAME xmlns:ns0="urn:DefaultNamespace"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <PRODUCTNUMBER xsi:type="xsd:string">45000</PRODUCTNUM
</ns0:GETPRODUCTNAME>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```


Пример 3.3. Контекст запроса SOAP

```
soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <soapenv:Body>
- <ns1:GETPRODUCTNAMEResponse xmlns:ns1="urn:DefaultNamespa
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <GETPRODUCTNAMEReturn xsi:type="xsd:string">Redbook Basic</
</ns1:GETPRODUCTNAMEResponse>
</soapenv:Body>
</soapenv:Envelope>
```

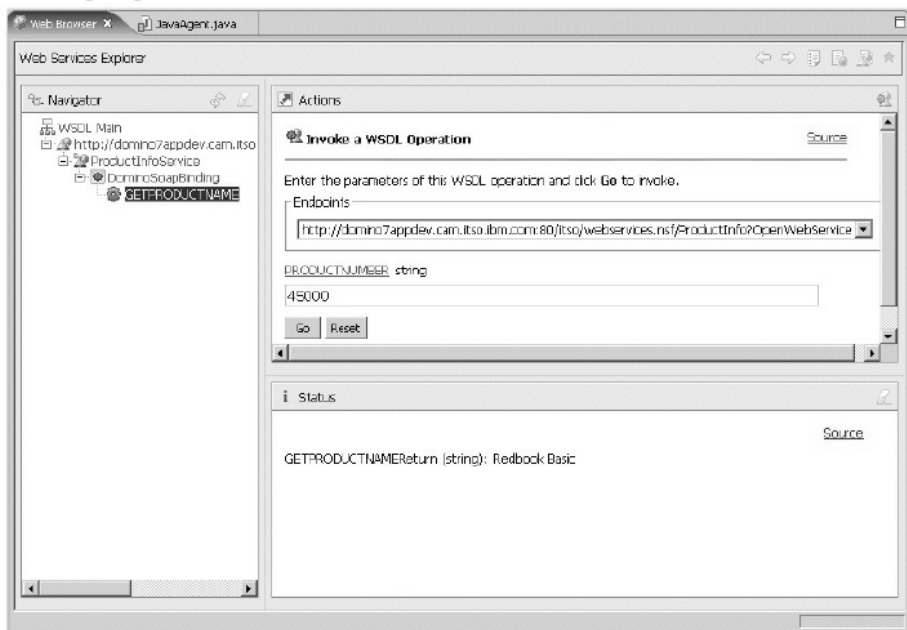
Пример 3.4. Контекст ответа SOAP, отправленный обратно с сервера

Рис. 3.10. Исходный код, представленный в панели Status

10. Наконец, чтобы увидеть разницу в формате XML, вернитесь в Lotus Domino Designer и измените формат сообщения SOAP на **Wrapped** в окне свойств **Advanced**, как показано на [рис. 3.11](#). Снова задействуйте Web-сервис GETPRODUCTNAME из Rational

Application Developer. Результат будет таким же, как в примерах 3.5 и 3.6.

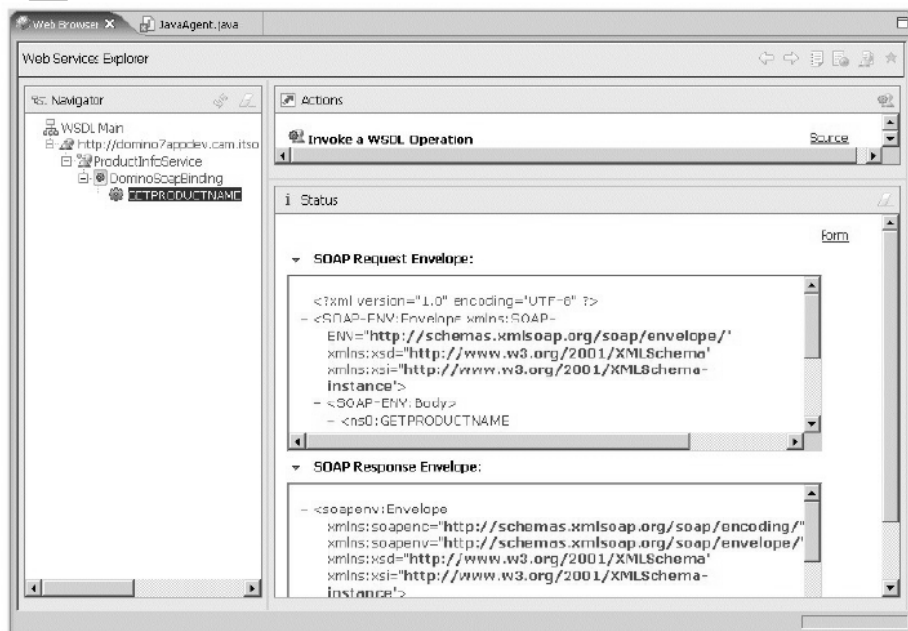


Рис. 3.11. Формат сообщения SOAP, установленный на Wrapped

```
<?xml version="1.0" encoding="UTF-8" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <SOAP-ENV:Body>
- <q0:GETPRODUCTNAME>
  <PRODUCTNUMBER>45000</PRODUCTNUMBER>
</q0:GETPRODUCTNAME>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Пример 3.5. Контекст запроса SOAP, в котором использован формат сообщения SOAP Wrapped

```
- <soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
- <soapenv:Body>  
- <ns1:GETPRODUCTNAMEResponse xmlns:ns1="urn:DefaultNamespac  
<GETPRODUCTNAMEReturn>Redbook Basic</GETPRODUCTNA  
</ns1:GETPRODUCTNAMEResponse>  
</soapenv:Body>  
</soapenv:Envelope>
```

Пример 3.6. Контекст ответа SOAP, в котором использован формат сообщения SOAP Wrapped

Совет Браузеру Web-сервисов необходимо получать обновленный документ WSDL от Web-сервиса. Всякий раз, когда документ WSDL изменяется, ваши потребители (включая браузер Web-сервисов) должны использовать самую последнюю версию документа WSDL. В противном случае операции могут быть выполнены не так, как вы ожидаете.

3.6 Создание сложного Web-сервиса в Lotus Domino Designer 7

В предыдущем примере было продемонстрировано совместное использование примитивных видов данных, таких как массивы, числа или (как в нашем случае) строки. Требования к приложению ITSO Electronics заключаются в предоставлении покупателям прямого доступа к информации о продуктах. Web-сервисы же позволяют нам раскрыть эти данные. Покупатели и поставщики могут увидеть цену определенного продукта или получить список всех доступных продуктов.

Для предоставления покупателям права доступа к информации о продуктах нам необходимо предоставить для совместного использования следующую информацию о продукте:

- Название продукта (`Product name`).
- Цена (`Price`).
- Номер (`Number`).
- Описание (`Description`).

В примере 3.7 приведены данные объекта `Product`. Этот объект будет использован в качестве *сложного типа данных* в Web-сервисе, над созданием которого мы будем работать в п. 3.5.1. В данном примере использован язык программирования Java.

```
public class Product
{
    private String name;
    private int number;
    private String description;
    private double price;
    public String getName(){ return name; }
    public int getNumber(){ return number; }
    public double getPrice(){ return price; }
    public String getDescription() { return description; }
    public void setName(String newName)
    {
        name = newName;
    }
    public void setNumber(int newNumber)
    {
        number = newNumber;
    }
    public void setDescription(String newDescription)
    {
        description = newDescription;
    }
    public void setPrice(double newPrice)
    {
        price = newPrice;
    }
}
```

Пример 3.7. Код `Product.java`

В п. 3.5.1, "Создание нового Web-сервиса", мы создавали Web-сервис, используя *примитивный тип данных*. В этом примере мы предпримем те же действия для создания Web-сервиса для объекта `Product`, который позволит обеспечить партнеров подробной информацией о

продуктах. В приведенных ниже действиях обобщается способ создания Web-сервиса.

1. Откройте базу данных ITSO Web Services в Domino Designer, затем Shared Code => Web Services (Общий код => Web-сервисы). Щелкните по кнопке New Web Service (Создать Web-сервис).
2. В окне свойств Web-service, в поле Name, введите ProductService, таким образом вы присвоите имя своему Web-сервису. В качестве класса PortType используйте ProductService.
3. В панели Options переключитесь на Java.
4. Щелкните по кнопке New Class (Создать класс), после чего замените код предоставляемым по умолчанию кодом, представленным в примере 3.8. Этот код называется ProductService.java.
5. Снова щелкните по кнопке New Class и вставьте код из примера 3.7. Этот код называется Product.java.
6. Сохраните Web-сервис.

```
import lotus.domino.*;
import lotus.domino.types.*;
import java.util.*;
public class ProductService
{
private Session session = WebServiceBase.getCurrentSession();
public Object[] getAllProducts()
{
Product[] result = null;
Database db = null;
View view = null;
Product product = null;
ArrayList tempList = new ArrayList();
try
{
db = getProductDb();
view = db.getView("vaPNumber");
Document doc2, doc = view.getFirstDocument();
```

```
while(doc != null)
{
product = getProductFromDocument(doc);
tempList.add(product);
doc2 = doc;
doc = view.getNextDocument(doc);
doc2.recycle();
}
view.recycle();
db.recycle();
} catch(NotesException e){e.printStackTrace();}
return tempList.toArray();
}

private Product getProductFromDocument(Document doc)
{
Product product = null;
try
{
String name = doc.getItemValueString("ProductName");
String description = doc.getItemValueString("ProductDescription");
double price = doc.getItemValueDouble("ProductPrice");
int productNumber = doc.getItemValueInteger("productNumber");
product = new Product();
product.setNumber(productNumber);
product.setName(name);
product.setPrice(price);
product.setDescription(description);
}
catch(Exception e)
{}
return product;
}

private Database getProductDb()
{
Database db = null;
try{
db = session.getDatabase("", "itso\\products.nsf");
} catch(NotesException e){}
return db;
```

```
}  
public Product getProduct(double productNumber)  
{  
    Product product = null;  
    try  
    {  
        Database db = getProductDb();  
        View view = db.getView("vaPNumber");  
        Document doc = view.getDocumentByKey(new Double(productNumber), true);  
        if(doc != null) {  
            product = getProductFromDocument(doc);  
            doc.recycle();  
            doc = null;  
        }  
        view.recycle();  
        db.recycle();  
    } catch (NotesException e) {}  
    return product;  
}  
}
```

Пример 3.8. ProductService.java

Мы можем использовать новый Web-сервис, названный ProductService из Rational Application Developer, как показано на рис. 3.12.

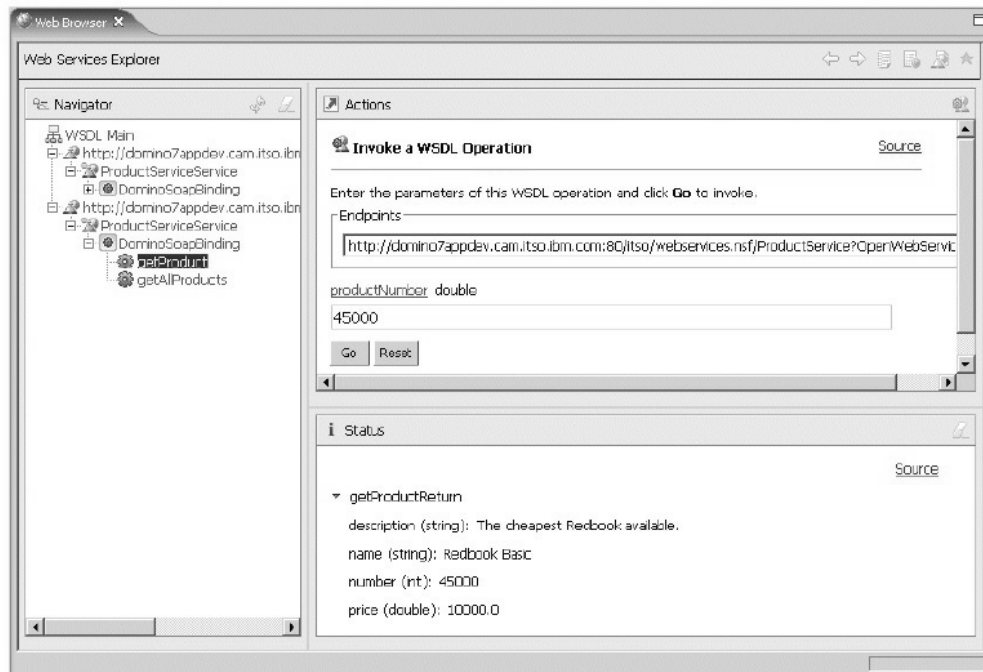


Рис. 3.12. Исполнение операции WSDL getProduct

Пример 3.9 выводит контекст ответа SOAP, полученный от сервера Lotus Domino для определенного номера продукта, который мы использовали для активирования операции WSDL getProduct.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <ns1:getProductResponse xmlns:ns1="urn:DefaultNamespace"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <getProductReturn href="#id0" />
    </ns1:getProductResponse>
  </multiRef xmlns:ns2="urn:DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id0" soapenc:
root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:
type="ns2:Product">
<description xsi:type="xsd:string">This is the basic Redbook offering.</descripti
```



```

<name xsi:type="xsd:string">Basic Redbook</name>
<number xsi:type="xsd:int">20501</number>
<price xsi:type="xsd:double">102.0</price>
</multiRef>
</soapenv:Body>
</soapenv:Envelope>

```

Пример 3.9. Контекст ответа SOAP, полученный от сервера

У Web-сервиса ProductService, который мы только что создали, есть и другая операция, отправляющая набор объектов продуктов в виде массива, которая называется `getAll-Products`. Пример 3.10 наглядно иллюстрирует этот процесс. Его сообщения SOAP имеют формат `RPC/encoded`.

```

- <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
- <soapenv:Body>
- <ns1:getAllProductsResponse xmlns:ns1="urn:DefaultNamespace"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
- <getAllProductsReturn xmlns:soapenc="http://schemas.xmlsoap.org/soap/
encoding/"
soapenc:arrayType="xsd:anyType[9]" xsi:type="soapenc:Array">
  <item href="#id0" />
  <item href="#id1" />
  <item href="#id2" />
  <item href="#id3" />
  <item href="#id4" />
  <item href="#id5" />
  <item href="#id6" />
  <item href="#id7" />
  <item href="#id8" />
</getAllProductsReturn>
</ns1:getAllProductsResponse>
- <multiRef xmlns:ns2="urn:DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id1" soapenc:
root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:

```

```

type="ns2:Product">
  <description xsi:type="xsd:string">Redpaper mid-tier line available at
newsagents and
department stores.</description>
  <name xsi:type="xsd:string">Redpaper Deluxe</name>
  <number xsi:type="xsd:int">10002</number>
  <price xsi:type="xsd:double">2299.0</price>
</multiRef>
- <multiRef xmlns:ns3="urn:DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id8" soapenc:
root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:
type="ns3:Product">
  <description xsi:type="xsd:string">The best Redbook money can buy.</
description>
  <name xsi:type="xsd:string">Redbook Plus</name>
  <number xsi:type="xsd:int">45500</number>
  <price xsi:type="xsd:double">50000.0</price>
</multiRef>
и так далее.....
- <multiRef xmlns:ns10="urn:DefaultNamespace"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" id="id6" soapenc:
root="0"
soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xsi:
type="ns10:Product">
  <description xsi:type="xsd:string">The ultimate Redbook experience.</
description>
  <name xsi:type="xsd:string">Redbook Supreme</name>
  <number xsi:type="xsd:int">20564</number>
  <price xsi:type="xsd:double">1000.0</price>
</multiRef>
</soapenv:Body>
</soapenv:Envelope>

```

Пример 3.10. Контекст ответа SOAP

3.7 Экспортирование документа WSDL

Документы WSDL экспортируют для того, чтобы предоставить

покупателю информацию о том, как использовать созданные Web-сервисы. Во многих случаях инструменты, с помощью которых осуществляется создание клиентов Web-сервисов, могут автоматически генерировать код из документа WSDL.

Для того чтобы экспортировать документ WSDL, выполните следующие действия:

1. Откройте базу данных ITSO Web Services в Domino Designer и выберите Shared Code => Web Services (Общий код => Web-сервисы).
2. Выберите ProductService и щелкните по кнопке Export WSDL (Экспортировать документ WSDL).
3. После этого будет сгенерирован файл WSDL в формате XML. В нашем сценарии мы использовали расширение .wsdl для того, чтобы опознать экспортированный файл WSDL, как видно из [рис. 3.13](#).

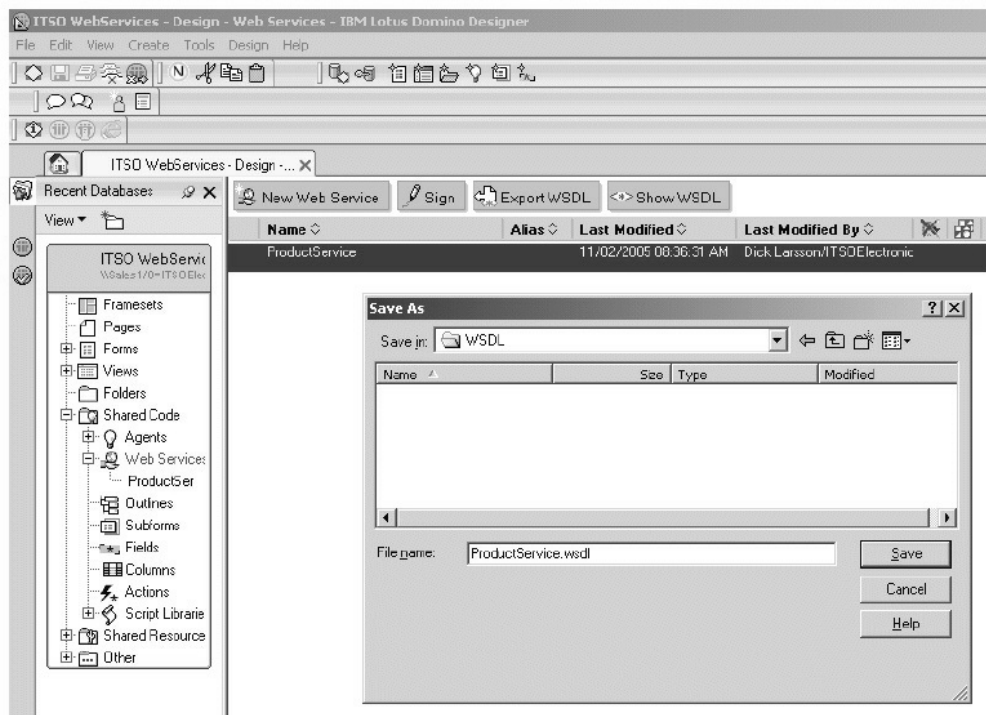


Рис. 3.13. Экспортирование документа WSDL

3.8 Импортирование документа WSDL

Если вы импортируете документ WSDL, описательный класс генерируется на основе содержания документа WSDL. Общедоступные прототипы методов, функций и подстановок в классе соответствуют операциям Web-сервиса, определенным для первого элемента `<port>` первого элемента `<service>` в документе WSDL. Другие классы или типы ссылаются на прототипы методов, которые, возможно, также будут сгенерированы на основании содержимого документа WSDL вместе с *общедоступными элементами* данных. Если вы измените прототип интерфейса, соответствующий документ WSDL также изменится, когда Web-сервис будет сохранен. Вы можете отслеживать подобные изменения с помощью свойства "Warn if the WSDL interface is modified" (Предупреждать, если интерфейс WSDL был изменен).

Бизнес-партнеры компании ITSO Electronics, а, возможно, и покупатели, в свою очередь, могут обеспечивать компанию документом WSDL, который она может использовать для создания Web-сервисов. Вы можете импортировать существующий документ WSDL, чтобы сгенерировать каркас Web-сервиса. Каркасный код соответствует описанию Web-сервиса. Затем вам необходимо добавить код реализации.

Для того чтобы импортировать документ WSDL, выполните следующее:

1. Откройте базу данных ITSO Web Services в Domino Designer, затем выберите Shared Code => Web Services.
2. Щелкните по кнопке Import WSDL и выберите файл WSDL, который хотите импортировать. В нашем сценарии мы импортировали файл, который экспортировали в предыдущем разделе (см. [рис. 3.14](#)).

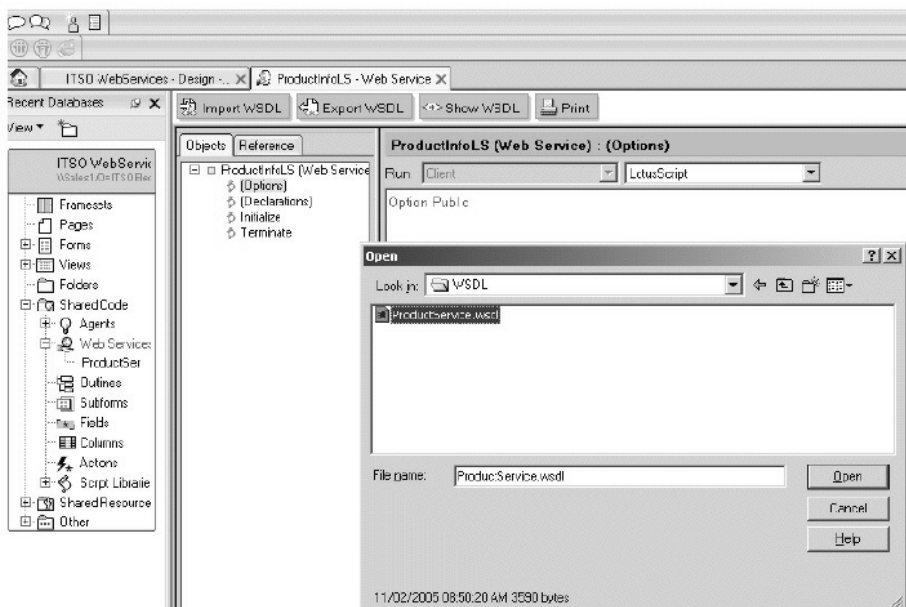


Рис. 3.14. Импортрование документа WSDL

3. В результате импортирования документа WSDL будет сгенерирован каркасный код, который в нашем сценарии сгенерирован на языке LotusScript, как показано в [примере 3.11](#). Затем мы открыли окно свойств Web Service, поставили галочку рядом с опцией Warn if the WSDL interface is modified ([рис. 3.15](#)).

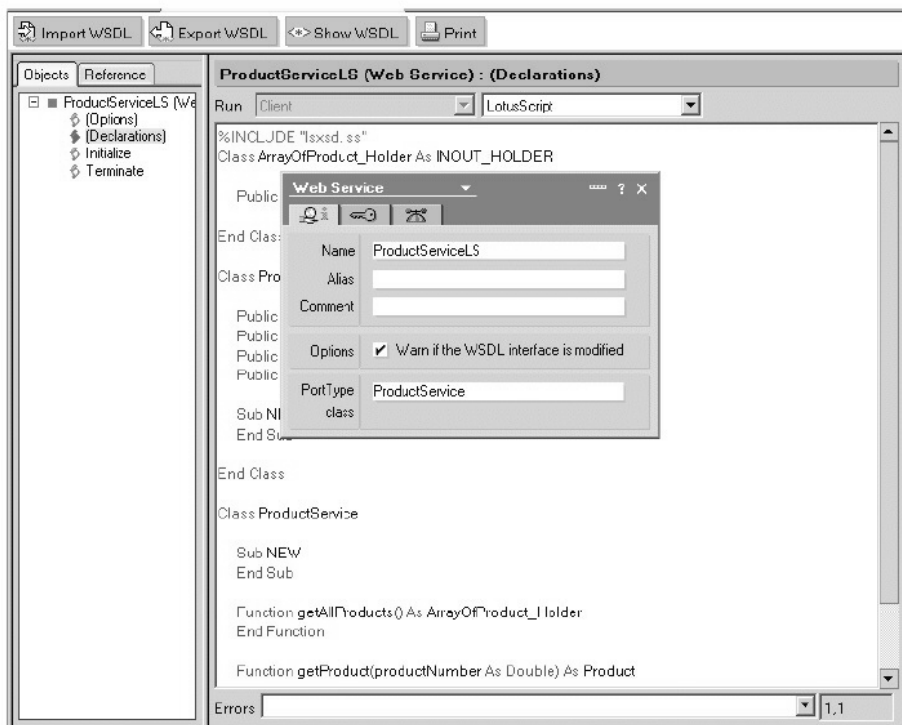


Рис. 3.15. Импортрование документа WSDL: окно свойств Web Service

```
%INCLUDE 'lsxsd.lss'
Class ArrayOfProduct_Holder As INOUT_HOLDER
Public Value() As Product
End Class
Class Product
Public description As XSD_STRING
Public name As XSD_STRING
Public number As Long
Public price As Double
Sub NEW
End Sub
End Class
Class ProductService
Sub NEW
End Sub
Function getAllProducts() As ArrayOfProduct_Holder
```

```
End Function
Function getProduct(productNumber As Double) As Product
End Function
End Class
```

Пример 3.11. Каркасный код, сгенерированный после импортирования документа WSDL

3.9 Работа с исключениями и ошибками в Web-сервисах

Операции также могут возвращать ошибки посредством подкласса ошибок, который соответствует некоторому сообщению `<wsdl:fault>`, выбранному для определенной операции в документе WSDL, или посредством прямого использования базового класса `WS_FAULT` (LotusScript), или класса `lotus.domino.types.Fault`, или класса `java.lang.Exception` (Java).

Замечание Любые ошибки в методе описания сервиса на языке LotusScript (т. е. любой основной класс или подкласс `WS_FAULT`) должны фигурировать в конце списка параметров метода.

Ошибки для метода описания сервиса на языке Java возникают в стандартных предложениях Java.

Для получения дополнительной информации о том, как работать с ошибками, используя язык LotusScript см. [п. 3.12](#), "Использование Web-сервисов с помощью LS2J".

3.10 Безопасность в Web-сервисах

Угрозы безопасности Web-сервисов, определенных в приложении ITSO Electronics, подразумевают угрозу всей системе, на которой базируется Web-сервис, приложению и инфраструктуре сети в целом. Для обеспечения защиты Web-сервисов необходим целый спектр механизмов защиты, основанных на языке XML, так как только благодаря им становится возможным решение проблем, касающихся

аутентификации, ролевого контроля доступа к базам данных, реализации принципов обеспечения безопасности, а также безопасности уровня сообщений, которая выполняет роль посредника.

В настоящий момент не существует общепринятых спецификаций безопасности Web-сервисов. Поэтому разработчики могут либо создавать службы, которые не используют подобных возможностей, либо разрабатывать специальные узкопрофильные программы, которые предположительно смогут решать проблемы функциональной совместимости.

Для применения Web-сервисов могут потребоваться механизмы защиты "точка-точка" или сквозные механизмы защиты, или оба механизма в зависимости от степени угрозы или риска. Традиционные механизмы безопасности, построенные на основе соединений "точка-точка", возможно, не смогут справиться с потребностями в сквозной защите Web-служб. Тем не менее, безопасность – это баланс между оцененным риском и затратами на противодействие. В зависимости от допускаемого риска использования защита, обеспечиваемая соединением "точка-точка", работающая на уровне переноса данных, позволяет обеспечить достаточными мерами безопасности.

В архитектуре приложения ITSO Electronics существует три фундаментальных принципа, относящихся к безопасности: ресурсы, которые должны быть защищены; механизмы, благодаря которым обеспечивается эта защита (т. е. политика защиты); и политики, которые представляют собой обрабатываемые компьютерами документы, которые, описывают ограничения на этих ресурсах. Для продолжения дискуссии о безопасности применения Web-сервисов обратитесь к руководству Web Service Architecture, которое доступно по следующему адресу в Интернете: ссылка: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>

Web-сервис Lotus Domino 7 обладает теми же возможностями защиты, что и агент. Уровень защиты Web-сервиса устанавливается в закладке *Security* (Безопасность) окна свойств Web service. В приведенном ниже списке описываются параметры, доступные в закладке *Security*.

- `Run as web user` (Запускать от лица Web-пользователя): код Java или LotusScript запускается с реальным именем пользователя, который запустил Web-сервис.
- `Run on behalf of` (Запускать от лица): код Java или LotusScript запускается на полномочиях определенного пользователя. Тем не менее, подписчик Web-сервиса должен обладать неограниченными правами на сервер или правом запускать Web-сервисы от лица любого пользователя.
- `Set runtime security level` (Установить уровень защиты выполнения) (один из перечисленных ниже):
 - `Do not allow restricted operations` (Не разрешать выполнение операций с ограниченным доступом);
 - `Allow restricted operations` (Разрешать выполнение операций с ограниченным доступом);
 - `Allow restricted operations with full administration rights` (Разрешать выполнение операций с ограниченным доступом с полными правами администрирования).
- `Default access for this web service` (Доступ по умолчанию к этому Web-сервису):
 - `All readers and above` (Все читатели и выше). Вы можете выбрать или снять этот параметр.
 - `Enumerated` (Перечисленные). Если вы убрали галочку рядом с параметром `All readers and above`, то с помощью этого параметра вы можете выбрать тех, кому вы хотите предоставить доступ по умолчанию.
- `Allow Public Access users to use this web service` (Разрешить пользователям открытого доступа применять этот Web-сервис). Если выбран этот параметр, пользователи, обладающие открытым доступом к документам в базе данных, получают доступ по умолчанию.

3.11 Использование Web-сервисов с помощью Java

На момент написания этой книги Lotus Domino 7.0 обеспечивал, хотя и не в полном объеме, использование Web-сервисов. Тем не менее, вспомогательным бизнес-требованием, предъявляемым к приложению

ITSO Electronics, является предоставление доступа к ценовым данным, располагающимся за границами Lotus Domino, что, в свою очередь, диктует необходимость использования Web-сервисов в этом приложении. В нашем сценарии компания ITSO Electronics решила применить программный продукт с открытым исходным кодом Apache SOAP, чтобы использовать Web-сервисы с помощью Java. Apache SOAP представляет собой реализацию стандарта SOAP в W3C. Он заменяет собой разработку IBM SOAP4J. Последующий, улучшенный проект Apache Axis доступен по следующему адресу: ссылка: <http://ws.apache.org/axis/>

Для того чтобы проиллюстрировать использование Web-сервисов, в нашем сценарии мы создали отдельную базу данных в приложении ITSO Electronics, которую назвали `WS Consumer`. В этой базе данных содержится агент, который будет использовать сложный Web-сервис, созданный нами ранее.

Замечание Web-сервисы могут быть добавлены к любой базе данных Lotus Notes и Domino. Но для достижения целей, поставленных в этой книге, все Web-сервисы добавлены к отдельным базам данных в приложении ITSO Electronics, которые называются `webservices.nsf` и `wsconsumer.nsf`.

Приведенная ниже последовательность действий наглядно иллюстрирует создание агента, использующего Web-сервис.

1. Загрузите самую последнюю версию Apache SOAP по следующей ссылке: ссылка: <http://ws.apache.org/soap/>
2. Извлеките загруженный файл во временную директорию, в нашем сценарии в качестве такой папки мы использовали папку `c:\soap`.
3. Экпортируйте сложный Web-сервис `ProductService` (находящийся в базе данных `Web Services`) в документ `WSDL`. В экспортированном документе `WSDL` содержится вся информация, необходимая для использования Web-сервиса.
4. Откройте документ `WSDL` в текстовом редакторе. Документ `WSDL` содержит информацию, описывающую *сложный тип данных*, возвращаемых этим Web-сервисом, как видно из примера 3.12. В нашем случае этот *сложный тип данных* называется

`Product`. Сложный элемент `Type` показывает свойства объекта `Product`, к которым относятся описание, название, номер и цена.

```
<wsdl:types>
  <schema targetNamespace="urn:DefaultNamespace" xmlns="http://www
XMLSchema"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
  <import namespace="http://schemas.xmlsoap.org/soap/encoding/">
  <complexType name="Product">
    <sequence>
      <element name="description" nillable="true" type="xsd:string"/>
      <element name="name" nillable="true" type="xsd:string"/>
      <element name="number" type="xsd:int"/>
      <element name="price" type="xsd:double"/>
    </sequence>
  </complexType>
  <complexType name="ArrayOfProduct">
    <complexContent>
      <restriction base="soapenc:Array">
        <attribute ref="soapenc:arrayType" wsdl:arrayType="impl:Product[]"/>
      </restriction>
    </complexContent>
  </complexType>
</wsdl:types>
```

Пример 3.12. Тип WSDL для `Product`

- Откройте новую базу данных `WS Consumer` в `Domino Designer`, выберите `Shared Code => Agents` (Общий код $\Rightarrow$ Агенты), после чего щелкните по кнопке `New agent` (Создать агент).
- В поле `Name` окна свойств `Agent` введите `ConsumeWS` и в разделе `Runtime` в выпадающем меню `Target` (Цель) выберите `None` (Отсутствует), как показано на [рис. 3.16](#).

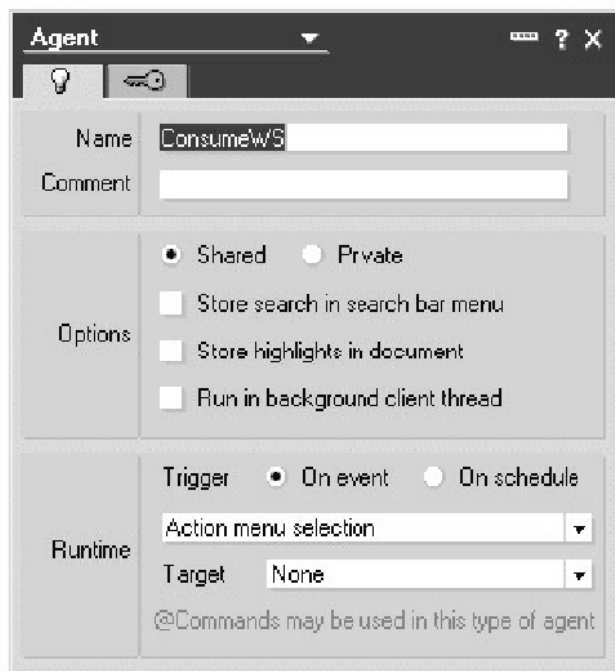


Рис. 3.16. Окно свойств ConsumerWS Agent

7. Закройте окно свойств Agent, затем выберите язык кода Java.
8. Нам необходимо создать класс `JavaBean`, представляющий сложный тип данных `Product`, определенный в документе WSDL. Для облегчения этой операции мы воспользуемся возможностью импортирования WSDL для автоматической генерации класса `JavaBean`. После этого нам необходимо выбрать `Shared Code => Web Service` (Общий код => Web-сервис) и щелкнуть по кнопке `New Web Service` (Создать Web-сервис).
9. Закройте окно свойств Web Service и убедитесь в том, что язык описания Web-сервиса установлен на Java. Щелкните по кнопке `Import WSDL` (Импортировать WSDL), затем выберите созданный ранее документ WSDL. В результате импортирования этого документа будут автоматически созданы классы `JavaBeansTM`.
10. Выделите и скопируйте код Java для `Product.java`.
11. Вернитесь к агенту и щелкните по кнопке `New Class` (Создать

класс). Затем замените класс `Untitled.java` скопированным кодом `Product.java`, так же, как это сделано в [примере 3.13](#).

12. Сохраните агент `ConsumeWS` и убедитесь в том, что он был успешно скомпилирован.

Щелкните по кнопке `Edit Project` (Изменить проект), откроется окно `Java Agent Files` (см. [рис. 3.17](#)). Перейдите к нужной директории библиотеки `SOAP LIB`, в нашем случае эта директория находилась по следующему пути: `c:\soap\soap-2_3_1\lib`. В качестве типа файла выберите `Archive` (Архивный), щелкните по кнопке `Add/Replace File(s)` (Добавить/Заменить файлы), далее нажмите `OK`.

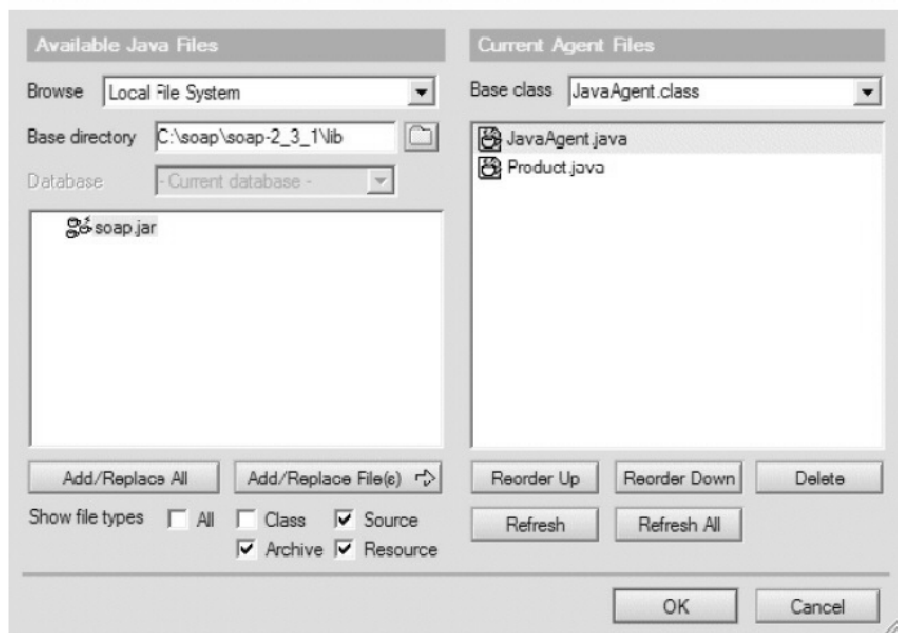


Рис. 3.17. Окно `Organize Java Agent Files`

13. Вам необходимо код `JavaAgent.java` по умолчанию заменить своим собственным, чтобы приступить к использованию Web-сервиса, который в нашем сценарии представлен в [примере 3.14](#). Убедитесь в том, что в классе `JavaAgent.java` есть переменные `URL` и `soapAction`. Эти переменные должны указывать на расположение вашего приложения.

14. Сохраните агент `ConsumeWS`.
15. Вернитесь к новому Web-сервису и закройте его без сохранения.
16. Откройте клиент Lotus Notes и выберите базу данных `WS Consumer`. Перейдите к `Actions => ConsumeWS` (Действия $\Rightarrow$ `ConsumeWS`) для того, чтобы запустить только что созданный агент. Агент отобразит набор объектов `Product`, используемых в Web-сервисе, в окне Java, как видно на [рис. 3.18](#).

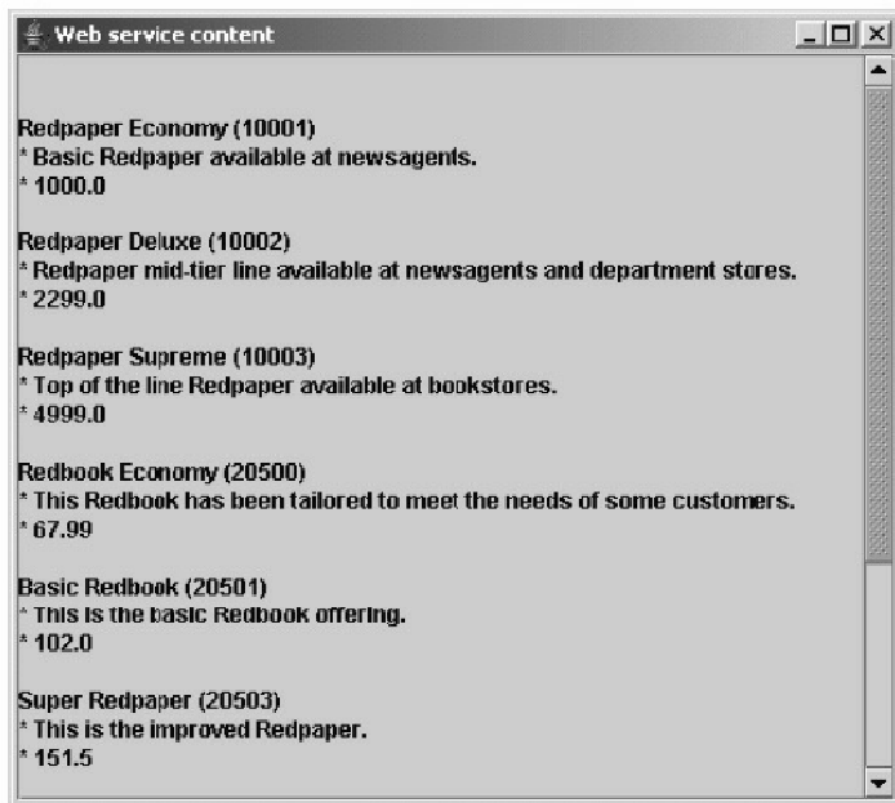


Рис. 3.18. Окно Java, в котором отображен результат работы Web-сервиса

Если окно не открылось, выберите `File => Tools => Show Java Debug Console` (Файл $\Rightarrow$ Сервис $\Rightarrow$ Показать консоль отладки Java) для получения дополнительной информации.

```
public class Product {  
    private java.lang.String description;
```

```
private java.lang.String name;
private int number;
private double price;
public Product() {
}
public java.lang.String getDescription() {
return description;
}
public void setDescription(java.lang.String description)
this.description = description;
}
public java.lang.String getName() {
return name;
}
public void setName(java.lang.String name) {
this.name = name;
}
public int getNumber() {
return number;
}
public void setNumber(int number) {
this.number = number;
}
public double getPrice() {
return price;
}
public void setPrice(double price) {
this.price = price;
}
private java.lang.Object __equalsCalc = null;
public synchronized boolean equals(java.lang.Object obj) {
if (!(obj instanceof Product)) return false;
Product other = (Product) obj;
if (obj == null) return false;
if (this == obj) return true;
if (__equalsCalc != null) {
return (__equalsCalc == obj);
}
__equalsCalc = obj;
```

```
boolean _equals;
_equals = true &&
((this.description==null && other.getDescription()==null) |
 (this.description!=null &&
  this.description.equals(other.getDescription())) &&
((this.name==null && other.getName()==null) ||
 (this.name!=null &&
  this.name.equals(other.getName())) &&
this.number == other.getNumber() &&
this.price == other.getPrice();
__equalsCalc = null;
return _equals;
}

private boolean __hashCodeCalc = false;
public synchronized int hashCode() {
if (__hashCodeCalc) {
return 0;
}
__hashCodeCalc = true;
int _hashCode = 1;
if (getDescription() != null) {
_hashCode += getDescription().hashCode();
}
if (getName() != null) {
_hashCode += getName().hashCode();
}
_hashCode += getNumber();
_hashCode += new Double(getPrice()).hashCode();
__hashCodeCalc = false;
return _hashCode;
}
}
```

Пример 3.13. Код класса Product

```
import lotus.domino.*;
import java.io.*;
import java.util.*;
import java.net.*;
```



```
import org.w3c.dom.*;
import org.apache.soap.util.xml.*;
import org.apache.soap.*;
import org.apache.soap.encoding.*;
import org.apache.soap.encoding.soapenc.*;
import org.apache.soap.rpc.*;
import javax.swing.*;
public class JavaAgent extends AgentBase
{
    public void NotesMain()
    {
        try
        {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();

            URL url = new URL("http://domino7appdev.cam.itso.ibm.com/itso/webservices.1
ProductService?WSDL");

            String soapAction = "http://domino7appdev.cam.itso.ibm.com/itso/webservices.ns
ProductService?WSDL";
            Call call = new Call();
            BeanSerializer beanSerializer = new BeanSerializer();
            SOAPMappingRegistry smRegistry = new SOAPMappingRegistry();
            smRegistry.mapTypes(Constants.NS_URI_SOAP_ENC, new QName(
                "urn:DefaultNamespace", "Product"), Product.class,
            beanSerializer,
                beanSerializer);
            String targetNamespace = "urn:DefaultNamespace";
            call.setSOAPMappingRegistry(smRegistry);
            call.setTargetObjectURI(targetNamespace);
            call.setMethodName("getAllProducts");
            call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
            Response resp = null;
            try
            {
                resp = call.invoke(url, soapAction);
            } catch (SOAPException e)
            {
```

```
e.printStackTrace();
```

```
System.err.println("SOAP Exception code: " + e.getFaultCode() + " message: " +
    return;
```

```
}
```

```
if (!resp.generatedFault())
```

```
{
```

```
    Parameter ret = resp.getReturnValue();
```

```
    Product[] products = (Product[]) ret.getValue();
```

```
    buildGUI(products);
```

```
}
```

```
else
```

```
{
```

```
    Fault fault = resp.getFault();
```

```
System.err.println("SOAP Fault, " + fault.getFaultCode() + " " + fault.
getFaultS
```

```
}
```

```
} catch (Exception e)
```

```
{
```

```
    e.printStackTrace();
```

```
}
```

```
}
```

```
private void buildGUI(Product[] products)
```

```
{
```

```
    StringBuffer sb = new StringBuffer();
```

```
    String newLine = "<br>";
```

```
    for(int i = 0; i < products.length; i++)
```

```
    {
```

```
        sb.append(newLine + newLine);
```

```
sb.append(products[i].getName() + " (" + products[i].getNumber() + ")" +
newLine);
```

```
    sb.append( " * " + products[i].getDescription());
```

```
    sb.append(newLine + " * " + products[i].getPrice());
```

```
}
```

```
JFrame frame = new JFrame("Web service content");
```

```
frame.getContentPane().add(new JScrollPane(new JLabel("<html>" + sb.
toString() + "</html>"))));
```

```
        frame.pack();  
        frame.setVisible(true);  
    }  
}
```

Пример 3.14. Код, необходимый для использования Web-сервиса с помощью Java

3.12 Использование Web-сервисов с помощью LS2J

В этом разделе мы расскажем о следующем:

- как использовать Web-сервис с помощью LotusScript;
- как с помощью LS2J обращаться к коду Java из агента LotusScript;
- как работать с исключениями или ошибками, возвращаемыми назад от Web-сервисов;
- об аутентификации и Web-сервисах.

3.12.1 Несколько слов о LS2J

LS2J представляет собой интерфейс, позволяющий переводить данные из типов данных Java в типы данных LotusScript и позволяет LotusScript выполнять методы объектов Java. Благодаря LS2J LotusScript получает способность создавать объекты Java так, будто они были родными для среды LotusScript.

В этом сценарии мы добавили агент LotusScript к базе данных WS Consumer. Агент использует код Java из библиотеки скриптов, для того чтобы использовать Web-сервис.

Web-сервис, который мы используем, способен выполнять одно из двух действий:

- возвращать имя текущего пользователя;
- возвращать объект `WS_FAULT`, для того чтобы проиллюстрировать, как Web-сервисы способны отправлять коды ошибок обратно к клиенту.

3.12.2 Создание Web-сервиса

Перечисленные ниже этапы показывают, как создать Web-сервис с помощью LS2J. В этом разделе описывается создание Web-сервиса, а в последующих речь пойдет о создании библиотеки скриптов Java и о том, как она используется в нашем агенте LotusScript через LS2J. Выполните приведенную ниже последовательность действий.

1. Откройте базу данных Web Services в Domino Designer.
2. Щелкните по кнопке `New Web Service`.
3. Убедитесь, что установлен язык кода LotusScript.
4. Перейдите к разделу `Declarations` и перепечатайте код, представленный в [примере 3.15](#).

```
Dim session As NotesSession
```

```
Class UserInfo
```

```
Function getUsername ( Fault1 As WS_FAULT)
```

```
Set session = New NotesSession
```

```
Dim doc As NotesDocument
```

```
Set doc = session.DocumentContext
```

```
If Cint(Rnd() * 10) > 5 Then
```

```
    getUsername = doc.remote_user(0)
```

```
Else
```

```
    Call Fault1.setFault(True) ' необходимо для ошибочной актива
```

```
    Call Fault1.setFaultString("SUCCESS, We send a FAULT on purp
```

```
End If
```

```
End Function
```

```
End Class
```

Пример 3.15. Код LS2J Web Service

5. Откройте окно свойств Web-сервиса и в поле `Name` и `PortType` введите `UserInfo`, как показано на [рис. 3.19](#).
6. Сохраните и закройте Web-сервис.

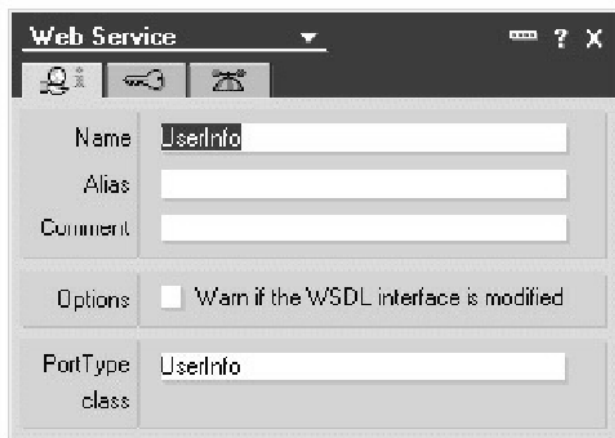


Рис. 3.19. Окно свойств LS2J Web Service UserInfo

3.12.3 Создание библиотеки скриптов, необходимой для использования Web-сервиса с помощью Java

Теперь нам необходимо создать библиотеку скриптов, содержащую класс Java, который будет использовать Web-сервис и сделает его доступным нашему агенту LotusScript через LS2J. Для того чтобы настроить класс Java, необходимо выполнить следующую последовательность действий:

1. Откройте базу данных WS Consumer в Domino Designer.
2. В навигационной панели выберите Shared Code (Общий код) и щелкните по кнопке New Java Library (Новая библиотека Java).
3. В открывшемся окне щелкните по кнопке Edit Project и выберите файл Apache soap.jar (см. [рис. 3.20](#)). Если у вас отсутствует файл Apache soap.jar, необходимо вернуться к [п. 3.11](#), "Использование Web-сервисов с помощью Java" для получения дополнительной информации о том, как загрузить Apache SOAP.
4. Сохраните библиотеку скриптов под именем ConsumeWSLibrary.
5. Замените код для библиотеки по умолчанию кодом, приведенным в [примере 3.16](#).

Важно Приведенные ниже строки из примера 3.16 должны быть скорректированы и соответствовать URL вашей базы данных:

```
URL url = new URL("http://domino7appdev.cam.itso.ibm.com/itso/  
webservicess.nsf/ProductService?WSDL");  
String soapAction = "http://domino7appdev.cam.itso.ibm.com/itso/  
webservicess.nsf/ProductService?WSDL";
```

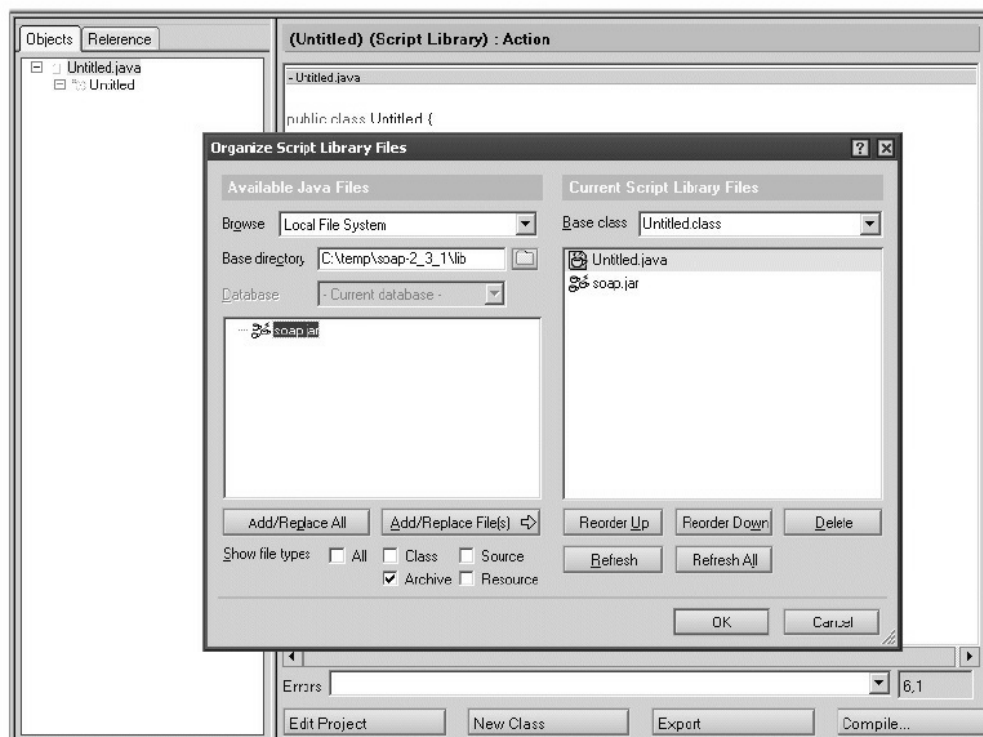


Рис. 3.20. Импорт архива Java, который будет использован с LS2J

```
import java.io.*;  
import java.util.*;  
import java.net.*;  
import org.w3c.dom.*;  
import org.apache.soap.util.xml.*;  
import org.apache.soap.*;  
import org.apache.soap.encoding.*;  
import org.apache.soap.encoding.soapenc.*;
```

```
import org.apache.soap.rpc.*;
import org.apache.soap.transport.http.SOAPHTTPConnection;
public class WSConsumer
{
    public String getValue(String userName, String password)
    {
        try
        {
            URL url = new URL("http://domino7appdev.cam.itso.ibm.com/itso/webservices.1
UserInfo?WSDL");

            String soapAction = "http://domino7appdev.cam.itso.ibm.com/itso/webservices.ns
UserInfo?WSDL";
            Call call = new Call();
            SOAPHTTPConnection hc = new SOAPHTTPConnection();
            hc.setUserName(userName);
            hc.setPassword(password);
            call.setSOAPTransport(hc);
            SOAPMappingRegistry smRegistry = new SOAPMappingRegistry();
            String targetNamespace = "urn:DefaultNamespace";
            call.setSOAPMappingRegistry(smRegistry);
            call.setTargetObjectURI(targetNamespace);
            call.setMethodName("getUserName");
            call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);
            Response resp = null;
            try
            {
                resp = call.invoke(url, soapAction);
            } catch (SOAPException e)
            {
                e.printStackTrace();
            }
            return ("SOAP Exception code: " + e.getFaultCode() + " message: " +
e.getMessage());
        }
        if (!resp.generatedFault())
        {
            Parameter ret = resp.getReturnValue();
            return ret.getValue() + "";
        }
    }
}
```

```

else
{
    Fault fault = resp.getFault();
    return ("SOAP Fault, " + fault.getFaultCode() + " " + fault.
getFaultString());
}
} catch (Exception e)
{
    e.printStackTrace();
}
return null;
}
}

```

Пример 3.16. Код Java, необходимый для использования Web-сервисов средой LS2J

3.12.4 Создание агента, используя язык LotusScript

Наконец нам необходимо создать агент LotusScript и использовать только что созданную библиотеку через LS2J.

1. В панели Navigation выберите Shared Code => Agents.
2. Щелкните по кнопке New Agent. В окне свойств нового агента, в поле Name, введите LotusScriptConsumeWS, а в выпадающем списке Target раздела Runtime выберите None. Закройте окно свойств.
3. Измените язык на LotusScript.
4. В Declarations замените код по умолчанию кодом, приведенным в [примере 3.17](#).
5. В Initialize замените код по умолчанию кодом из [примера 3.18](#).

Замечание Скорректируйте строку value =
myObject.getValue("The UserName", "The
Password"), введя правильное имя пользователя и пароль.

```

Option Public
Use "ConsumeWSLibrary"

```


UseIsx "*javacon"

Пример 3.17. Код Declaration для агента LotusScript LS2J

```
Sub Initialize
Dim mySession As JavaSession
Dim myClass As JavaClass
Dim myObject As JavaObject
Set mySession = New JavaSession ()
Set myClass = mySession.GetClass("WSConsumer")
Set myObject = myClass.CreateObject
Dim value As String
value = myObject.getValue("The UserName", "The Password")
Msgbox value
End Sub
```

Пример 3.18. Код Initialize для агента LotusScript LS2J

6. Сохраните и закройте агент.
7. В клиенте Lotus Notes запустите только что созданный агент. Если вы увидите сообщение, которое представлено на [рис. 3.21](#), то оно, скорее всего, вызвано тем, что вы не обеспечили код точным именем пользователя и паролем.

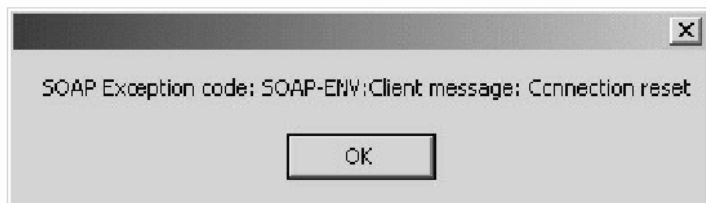


Рис. 3.21. Сообщение об ошибке, возникшей при использовании Web-сервиса с помощью LS2J

Если агент смог использовать ваш Web-сервис, вы увидите сообщение, представленное на [рис. 3.22](#).

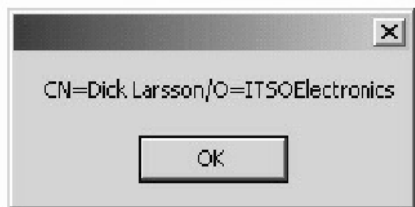


Рис. 3.22. Web-сервис был успешно использован с помощью LS2J

3.12.5 Аутентификация и Web-сервисы

В Lotus Notes и Domino 7 Web-сервис обладает теми же возможностями защиты, что и агент. В [примере 3.19](#) мы использовали базовую аутентификацию HTTP, для того чтобы защитить наш Web-сервис. Apache SOAP позволяет нам отправлять имена пользователей и пароли в виде SOAP-запросов, добавляя экземпляр класса `SOAPHTTPConnection` к объекту `Call`.

```
Call call = new Call();  
SOAPHTTPConnection hc = new SOAPHTTPConnection();  
hc.setUserName(userName);  
hc.setPassword(password);  
call.setSOAPTransport(hc);
```

Пример 3.19. Базовая аутентификация с помощью Apache SOAP

3.13 Какой стиль WSDL мне следует использовать?

В документе WSDL содержится XML-код, который описывает Web-сервис. Привязка WSDL описывает то, как сервис связан с протоколом обмена сообщениями. Привязка WSDL SOAP может быть осуществлена либо при помощи вызова удаленных процедур (RPC), либо при помощи документной привязки. Привязка SOAP также может употребляться как в кодированном виде, так и в литеральном виде. Таким образом, у разработчика приложений Lotus Domino есть четыре типа/моделей использования на выбор:

- вызов удаленных процедур/кодированный (RPC/encoded);

- вызов удаленных процедур/литеральный (RPC/literal);
- документальный/литеральный (Document/literal);
- заключенный в оболочку (Wrapped).

Терминология "RPC против документального" может сбивать с толку, так как эти термины подразумевают, что тип RPC следует использовать для программируемых моделей RPC и что документальный тип нужно использовать для документальных программируемых моделей или для программируемых моделей обмена сообщениями. Тем не менее, в данном случае все эти заблуждения бессмысленны, так как тип не имеет ничего общего с программируемыми моделями. Типы только диктуют то, как переводить привязку WSDL в сообщение SOAP. Подобно этому, типы кодированный и литеральный имеют значение только в привязке WSDL-к-SOAP.

¹⁾ С Web-сайта ссылка: <http://www.w3.org/Submission/2000/05/>.

Применение новых элементов разработки в приложении ITSO Electronics

В лекции описаны основы работы с новыми элементами разработки и рассмотрены дополнения к классам LotusScript, COM, OLE, Java и CORBA

Новые функции и возможности Lotus Domino Designer 7 основаны на более тесной интеграции с развивающимися стандартами Web, увеличении функциональной совместимости с другими стандартами программного обеспечения IBM и упрощении использования приложений разработчиками и пользователями.

В этой лекции мы проиллюстрируем, как новые функции Lotus Domino 7 используются для улучшения приложения ITSO Electronics. Рассмотрим следующие функции:

- команды контекстного меню;
- свойство формы *AutoSave*;
- общие столбцы;
- расширенные настройки для представлений и папок;
- дополнения к языку формул;
- дополнения к классам LotusScript, COM и OLE;
- дополнения к классам Java и CORBA.

4.1 Команды контекстного меню

У разработчиков приложений теперь появилась возможность добавлять действия в меню, выпадающее по нажатию правой кнопки мыши (иначе их называют контекстными) в представлениях и папках. По правому щелчку мыши открывается быстрый доступ к функциям, к которым пользователи обращаются регулярно. Такой способ воздействует на выбранные документы в представлении и технически ничем не отличается от щелчка по кнопке в панели *Action*. С точки зрения пользовательского интерфейса, эта функция полезна, когда пользователь предпочитает обрабатывать одиночный документ на основе информации, имеющейся в представлении, например при отправке файла в часто используемую папку, удалении или выделении фрагмента

текста. Добавление действий в контекстное меню может быть использовано как в качестве альтернативы, так и вместе с действиями в панели Action и меню Action. В контекстное меню могут быть добавлены и новые, и уже существующие действия.

В приложении ITSO Electronics в представлении "Sales Activity By Customer" существует командная кнопка Create New Sales Activity. Для того чтобы увеличить эффективность использования приложения, мы обновили параметр Display окна свойств Action и отметили галочкой свойство Include in right mouse button menu (Включить в контекстное меню) (рис. 4.1).

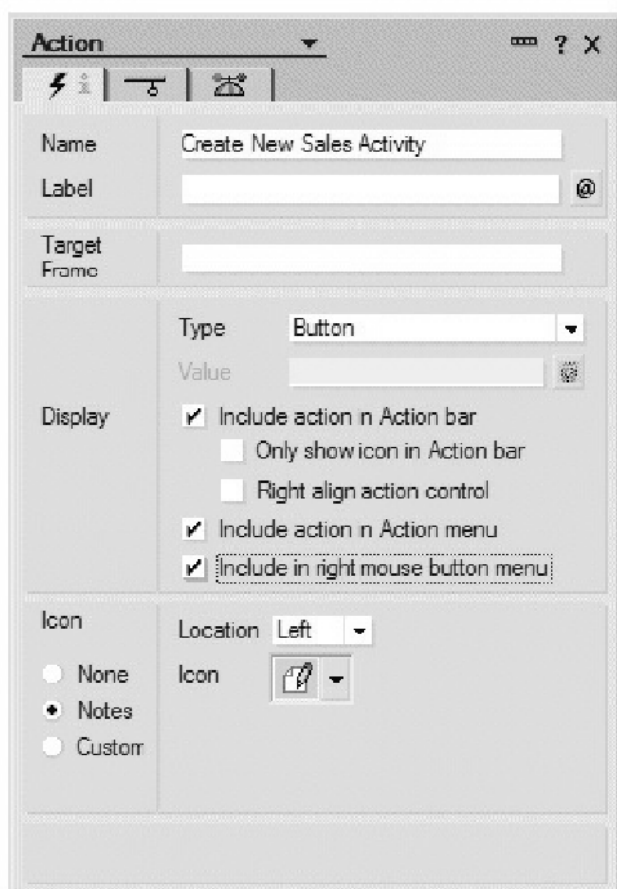


Рис. 4.1. Параметр Include in right mouse button menu

После щелчка правой кнопкой мыши в представлении появится контекстное меню, в котором будет только что добавленное действие (рис. 4.2). Мы также добавили в контекстное меню и другие действия.

Замечание Контекстное меню не доступно для пользователей, ограниченных в возможностях. Все действия, выполняемые в контекстном меню, должны быть также доступны в меню Actions.

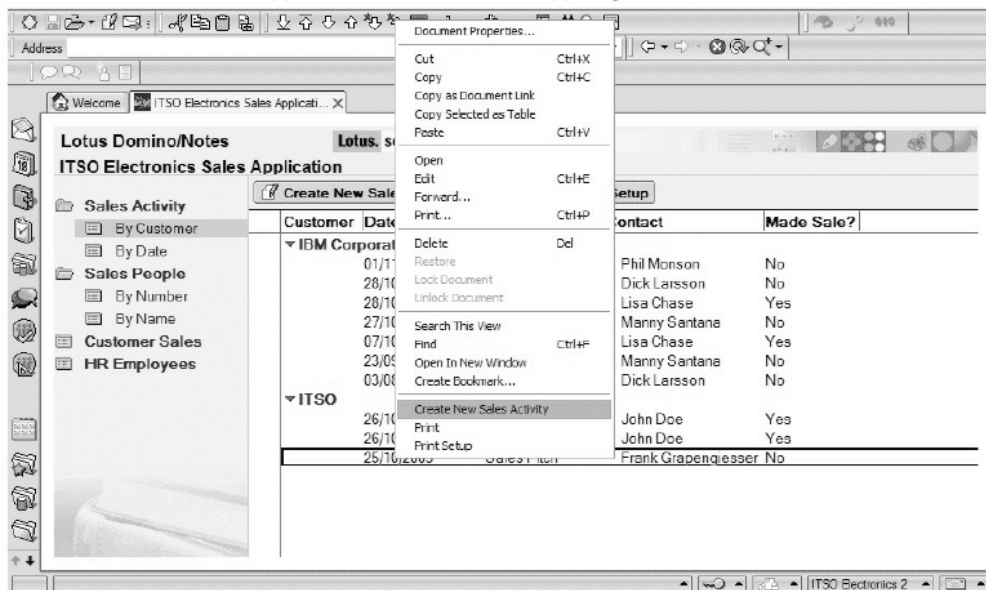


Рис. 4.2. Действие включено в контекстное меню

4.2 Свойство формы AutoSave

AutoSave в клиенте Lotus Notes – это функция, с помощью которой осуществляется регулярное автоматическое сохранение документов Lotus Notes в локальную базу через определенные интервалы времени, определяемые пользователем. В случае сбоя программы, рабочей станции или перебоев в питании пользователь сможет восстановить документ, который редактировал.

4.2.1 Включение поддержки функции AutoSave в клиенте Lotus Notes

При установке Lotus Notes 7 функция *AutoSave* не включена по умолчанию. Для ее включения необходимо выбрать соответствующий параметр в Lotus Notes Client User Preferences (Параметры пользователя клиента Lotus Notes).

1. Выберите File => Preference => User Preferences (Файл => Параметры => Параметры пользователя).
2. На вкладке Basics в параметрах *Startup Options* (Параметры запуска) активируйте опцию *AutoSave every 15 minutes* (Производить автосохранение каждые 15 минут) (см. [рис. 4.3](#)).
3. (Дополнительно) увеличьте или уменьшите интервал выполнения *AutoSave*.
4. Щелкните по кнопке ОК.

Замечание По умолчанию *AutoSave* производится каждые 15 минут. Интервал выполнения этой операции находится в пределах от 1 до 999 минут.

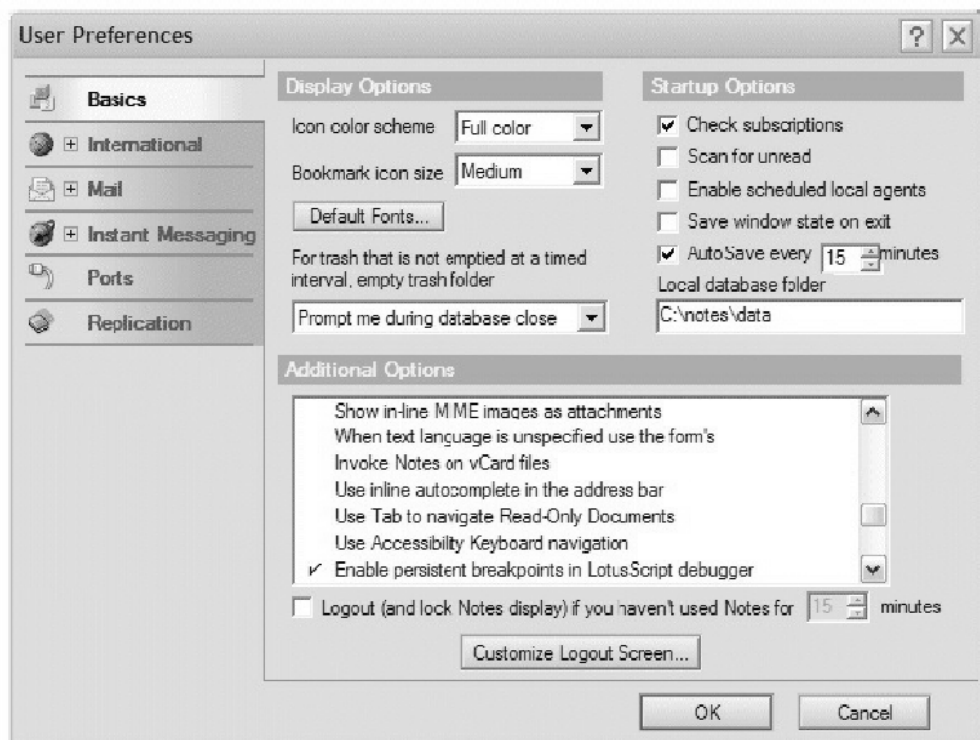


Рис. 4.3. Активация функции *AutoSave* в User Preferences

Параметры *AutoSave* в окне User Preferences соответствуют трем переменным в файле NOTES.INI. Эти переменные приведены в табл. 4.1.

Таблица 4.1. Переменные в файле NOTES.INI

Переменная	Параметр в User Preference
AUTO_SAVE_ENABLE=n	Функция <i>AutoSave</i> отключена, когда n=0
AUTO_SAVE_INTERVAL=<mins>	<mins> устанавливает временные интервалы между выполнением <i>AutoSave</i>
AUTO_SAVE_USER, <Abbreviated User Name>=<Database name relative to data directory>	Устраняет конфликты имен в базах данных с функцией <i>AutoSave</i>

4.2.2 Активирование *AutoSave* для формы

В Lotus Domino Designer использованы преимущества новой функции *AutoSave*, заключающиеся в возможности ее активизации в форме, находящейся в приложении Lotus Notes. Если пользователи активируют функцию *AutoSave* в параметрах Lotus Notes, как показано в предыдущем разделе, документы, которые они создают или редактируют, будут сохранены в отдельную локальную зашифрованную базу данных с помощью формы, в которой активированы *AutoSave*. Для того чтобы активировать функцию *AutoSave* для формы, откройте форму в Domino Designer и поставьте галочку рядом с опцией Allow Autosave (Разрешить выполнение Autosave) в окне свойств Form (см. рис. 4.4). Если функция *AutoSave* активирована в клиенте Lotus Notes, ее выполнение каждый раз будет отображаться в строке состояния.

The screenshot shows the 'Form' properties dialog box in Lotus Domino. The 'Name' field contains 'Sales Activity'. The 'Type' is set to 'Document'. In the 'Display' section, 'Include in menu' and 'Include in Search Builder' are checked, and 'Include in Print' is unchecked. The 'Versions' section shows 'Versioning' set to 'None' and 'Create versions' as an empty field. The 'Options' section has 'Allow Autosave' checked, while 'Default database form', 'Store form in document', 'Disable Field Exchange', 'Automatically refresh fields', 'Anonymous Form', 'No Initial Focus', 'No Focus On F6', 'Sign Documents that Use This Form', 'Render pass through HTML in Notes', and 'Do not add field names to field index' are unchecked. The 'Conflict Handling' section has 'Create Conflicts' selected.

Рис. 4.4. Параметр Allow AutoSave в окне свойств Form

Разработчикам приложений следует убедиться в том, что функция *AutoSave* при активации в формах приложения корректно работает с их приложением. Во многих приложениях Lotus Notes и Domino присутствует код, создающий элементы по событиям *Save* (Сохранение) и *Postsave* (Постсохранение) или производящий проверку достоверности в момент загрузки. Из-за того что коды проверки достоверности и события формы не выполняются во время автоматического сохранения документа, автоматически сохраненный документ может не загрузиться после восстановления. Возможно, потребуется добавить код в событие *Queryopen*, чтобы установить

значения по умолчанию для конкретных элементов, чтобы гарантировать правильную загрузку документов после восстановления.

Элемент `$AutosaveRecovery` можно использовать для работы с восстанавливаемыми документами. `$AutosaveRecovery` доступен для восстановленных документов до события `PostOpen`, а после этого события он удаляется. Например, форма может содержать код `Queryopen`, предотвращающий открытие документа в режиме редактирования, если поле `Status` не эквивалентно полю `Draft`. Если пользователь изменил статус на "Public" и произошло автоматическое сохранение, а после этого произошел сбой в работе компьютера, пользователю все еще необходимо открыть этот документ в режиме редактирования, несмотря на то, что значение поля `Status` кажется некорректным. Обратите внимание на следующий код:

```
Sub Queryopen(Source As Notesuidocument, Mode As Integer, Isnewdoc As V
Continue As
Variant)
Dim doc As NotesDocument
If Mode And Not Isnewdoc Then
Set doc = Source.Document
If doc.Status(0) <> "Draft" Then
Continue = False
MessageBox "Only Draft documents may be edited.", _
MB_ICONSTOP, "Edit Error"
End If
End If
End Sub
```

Вместо этого можно написать:

```
If doc.Status(0) <> "Draft" And Not doc.HasItem("$AutosaveRecovered ") The
```

Элемент `$DontAutosave` можно использовать для отключения функции `AutoSave` в данном конкретном документе. Функция `AutoSave` может быть снова активирована путем удаления `$DontAutosave`.

4.2.3 Восстановление документов, сохраненных с помощью AutoSave

Автоматически сохраненные документы могут быть восстановлены либо сразу после загрузки, либо в любое другое время после запуска. При выполнении запуска после сбоя системы или перебоя в питании, после аутентификации пользователю предлагается восстановить несохраненный документ (см. [рис. 4.5](#)).

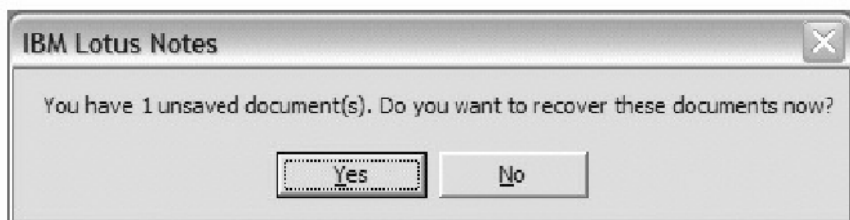


Рис. 4.5. Предложение восстановить несохраненный документ

Если вы щелкнете по кнопке Yes (Да), откроется окно Recover Unsaved Documents (Восстановить несохраненные документы), в котором перечислены все документы, которые могут быть восстановлены (см. [рис. 4.6](#)).

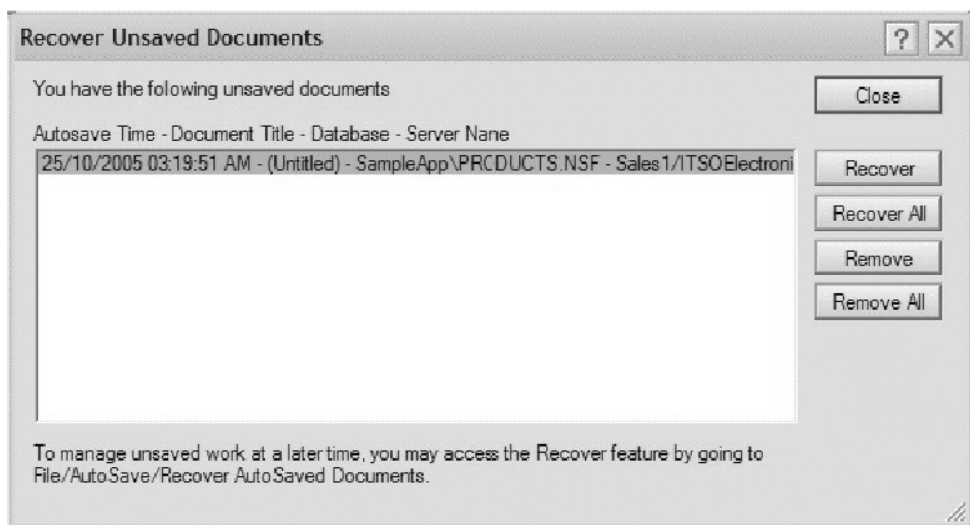


Рис. 4.6. Окно Recover Unsaved Documents

В табл. 4.2 описаны функции каждой из четырех кнопок окна, представленного на рис. 4.6.

Таблица 4.2. Варианты, предлагаемые в окне Recover Unsaved Documents

Вариант	Описание
Recover (Восстановить)	Восстановление выбранных документов и удаление их из списка.
Recover All (Восстановить все)	Восстановление всех документов и удаление их из списка; в этом случае не нужно подтверждать выполнение операции для каждого элемента
Remove (Удалить)	Удаление выбранных документов из базы данных <i>AutoSave</i>
Remove All (Удалить Все)	Удаление всех документов из базы данных <i>AutoSave</i>

Если пользователь щелкнул по кнопке No, то восстановление документов, сохраненных автоматически, можно осуществить позже, выбрав File => AutoSave => Recover AutoSaved Documents (Файл => Автоматическое сохранение => Восстановить документы, сохраненные автоматически). После этого появится окно Recover Unsaved Documents, и пользователь сможет восстановить или удалить документы описанным выше способом.

Документ не может быть восстановлен, если не найдена реплика базы данных документов. Он остается в базе данных *AutoSave* "как есть", и позже можно снова его восстановить. Если документ был модифицирован с момента автоматического сохранения, появится соответствующее сообщение и пользователю будет предложено либо переписать документ, используя сохраненный вариант, либо сохранить внесенные изменения. Если пользователь решит восстановить документ, полученный в результате автоматического сохранения, документ будет открыт в Lotus Notes в режиме редактирования. *AutoSave* не будет автоматически сохранять документ от лица пользователя после восстановления. Если документ восстановлен из базы данных *AutoSave*, он также удаляется из базы данных, что

позволяет поддерживать приемлемые размеры базы данных *AutoSave*.

4.2.4 Использование функции *AutoSave* вручную

Замечание Этот вариант работает только в том случае, если в форме активировано автоматическое сохранение.

Иногда пользователю, редактирующему документ, требуется немедленно его сохранить. Текущий документ можно сохранить в базу данных *AutoSave*, выбрав *File => AutoSave => Autosave Now* (Файл $\Rightarrow$ Автоматическое сохранение $\Rightarrow$ Выполнить автоматическое сохранение немедленно).

4.2.5 Получение доступа к базе данных *AutoSave*

База данных *AutoSave* находится в пользовательской директории *Data*. После создания база данных *AutoSave* именуется файлы так же, как файлы почты. К названию базы данных добавляется приставка *as_*. В нашем сценарии у пользователя *Manny Santana* база данных *AutoSave* будет называться: *as_msantana.nsf*. Если база данных с таким именем уже существует, то к названию будет добавлена цифра 1 и далее, до тех пор, пока не будет создано уникальное имя.

Учитывая особенности присвоения имен в базе данных *AutoSave* будут возникать конфликты с такими именами пользователей, как *John Doe*, *Jane Doe*, *James Doe* и т. д. Чтобы быть уверенными в том, что *AutoSave* соответствует текущему пользователю, после создания базы данных необходимо проверить пользователя, указанного в файле *NOTES.INI*:

```
AUTO_SAVE_USER,<Abbreviated User Name>=<Database name relative to directory>
```

В нашем приложении этот элемент будет добавлен к файлу *NOTES.INI* пользователя *Manny Santana*:

AUTO_SAVE_USER,Manny Santana/ITOSElectronics=as_msantana.nsf

Когда пользователь запускает Lotus, происходит проверка файла NOTES.INI на наличие элементов, относящихся к базе данных *AutoSave*. Если такие элементы найдены, название базы, заданное в элементе, используется в качестве названия базы данных *AutoSave*. Если пользователь не обладает доступом к заданной базе данных *AutoSave*, он получит сообщение об ошибке и функция *AutoSave* будет отключена. Если заданной базы данных не существует, будет создана новая база данных, при этом будет использовано имя, заданное в элементе. Если соответствующий элемент с именем пользователя отсутствует в файле NOTES.INI, то он в нем создается, соответственно создается база данных с учетом правил присвоения имен, описанных выше.

4.3 Общие столбцы (Shared Columns)

Программа Lotus Domino Designer 7 предоставляет возможность создания общих столбцов, используемых в представлениях и папках. Общий столбец – это отдельный элемент дизайна, однако, будучи вставленным в представление, он становится его частью. Разрабатываемый общий столбец обладает теми же свойствами и внешним видом, что и неотделяемый столбец в представлении и папке. Кроме того, общий столбец обладает специфичными свойствами. В окне свойств *Shared Column* (см. [рис. 4.7](#)), можно задать название столбца (обязательно), псевдоним (не обязательно) и комментарий (не обязательно).

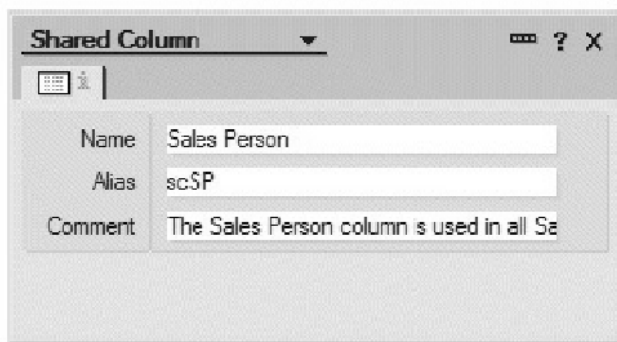


Рис. 4.7. Окно свойств Shared Column

Представление или папка, содержащие общий столбец, обратно совместимы для доступа из Всемирной сети и Lotus Notes. Общие столбцы могут быть использованы в DB2 Query View, о чем уже упоминалось в п. 2.3 "Query Views". Общие столбцы, подобно общим полям, не являются многоязычными элементами.

Использование общих столбцов позволяет разработчикам увеличить производительность, за счет создания общих столбцов, вставляемых во множество представлений и папок, находящихся в одной базе данных. То есть нет необходимости в создании одного и того же столбца несколько раз. Благодаря этой новой функции можно распространять изменения на все представления, используя один и тот же столбец.

4.3.1 Создание общего столбца

Чтобы создать новый общий столбец, выберите `Column` в меню `Shared Code`. Щелкните по кнопке `New Shared Column` (Создать общий столбец) или выберите `Create => Design => Shared Column` (Создать $\Rightarrow$ Разработка $\Rightarrow$ общий столбец). Также можно копировать и вставлять существующий разрабатываемый общий столбец.

Чтобы создать новый общий столбец, основываясь на разработке существующего столбца в представлении, откройте представление, выделите столбец, затем выберите `Create => Copy As Shared Column` (Создать $\Rightarrow$ Скопировать как общий столбец). В качестве альтернативы можно выделить столбец, открыть контекстное меню и выбрать `Copy As Shared Column` (см. [рис. 4.8](#)).

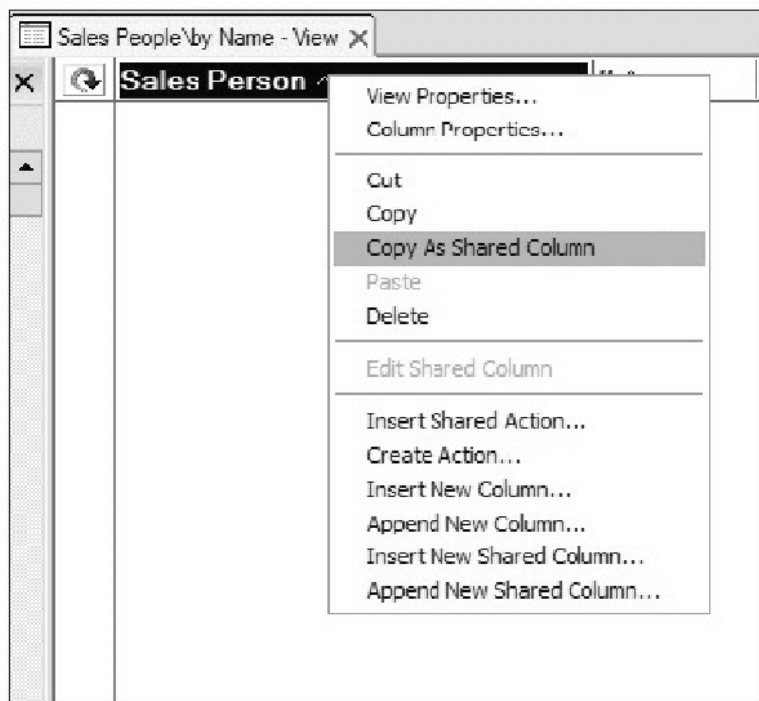


Рис. 4.8. Использование контекстного меню для создания нового общего столбца

После этого появится окно, в котором надо ввести название нового общего столбца (см. [рис. 4.9](#)). В результате будет создан новый общий столбец, а текущий столбец останется без изменений. Существующий столбец полностью независим от только что созданного общего столбца, поэтому если пользователю захочется унаследовать изменения дизайна из нового общего столбца, ему необходимо удалить существующий столбец в представлении и вставить новый общий столбец, чтобы заменить тот неотделяемый столбец.

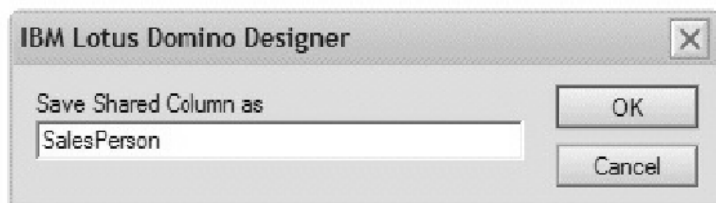


Рис. 4.9. Присвоение имени новому общему столбцу

В приложении ITSO Electronics существует представление "Sales People By Name", в котором есть столбец Sales Person (Торговый агент). Нам необходимо отобразить такую же информацию в представлении "Sales People By Number". Следовательно, столбец Sales Person будет использован для создания нового общего столбца, который мы назовем Sales Person. Представления, в которых использованы общие столбцы, могут быть открыты в предыдущих версиях клиента Lotus Notes.

4.3.2 Использование общего столбца

Для использования общего столбца в представлении откройте представление и выберите Create => Insert shared column (Создать => Вставить общий столбец) или Create => Append shared column (Создать => Добавить общий столбец). Выберите общий столбец из представленного в окне списка (рис. 4.10). Отметьте галочкой опцию Use Formula Only (Использовать только язык формул), если хотите заменить свойства отображения колонки, такие как шрифт и расположение колонки в представлении.

Не выбирайте этот параметр, если хотите зафиксировать свойства отображения. *Programmatic Name* (Программное имя) (в окне Properties => закладка Advanced) вставленного/присоединенного столбца не наследуется из общего столбца, вне зависимости от параметра опции Use Formula Only ; следовательно, это значение всегда может быть изменено. Тем не менее, значение столбца (формула) не может быть изменено.

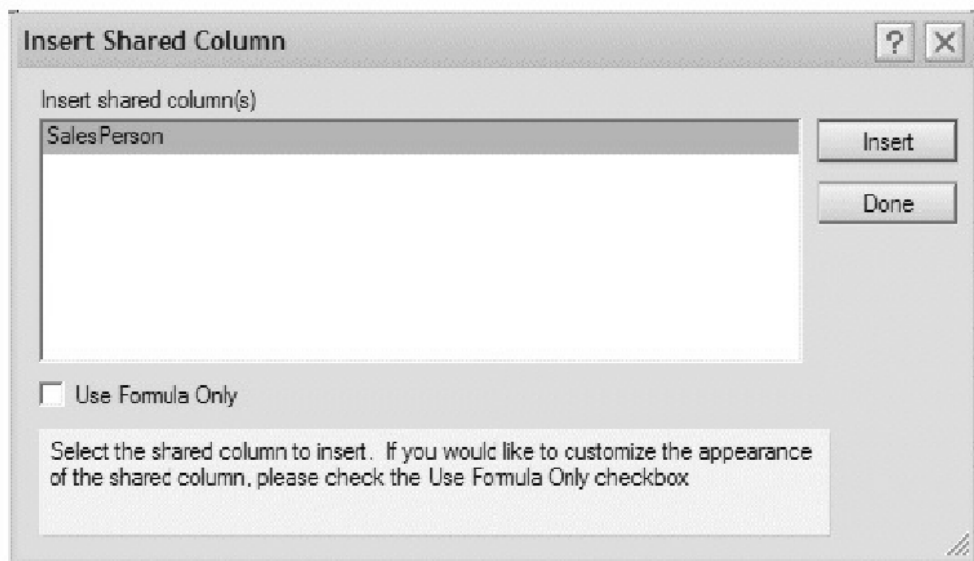


Рис. 4.10. Окно Insert Shared Column

Чтобы использовать общий столбец `Sales Person` в представлении "Sales People By Number", мы открыли представление в Domino Designer и вставили общий столбец, выполнив последовательность действий, описанную выше. На [рис. 4.11](#) представлено обновленное представление со вставленным общим столбцом.

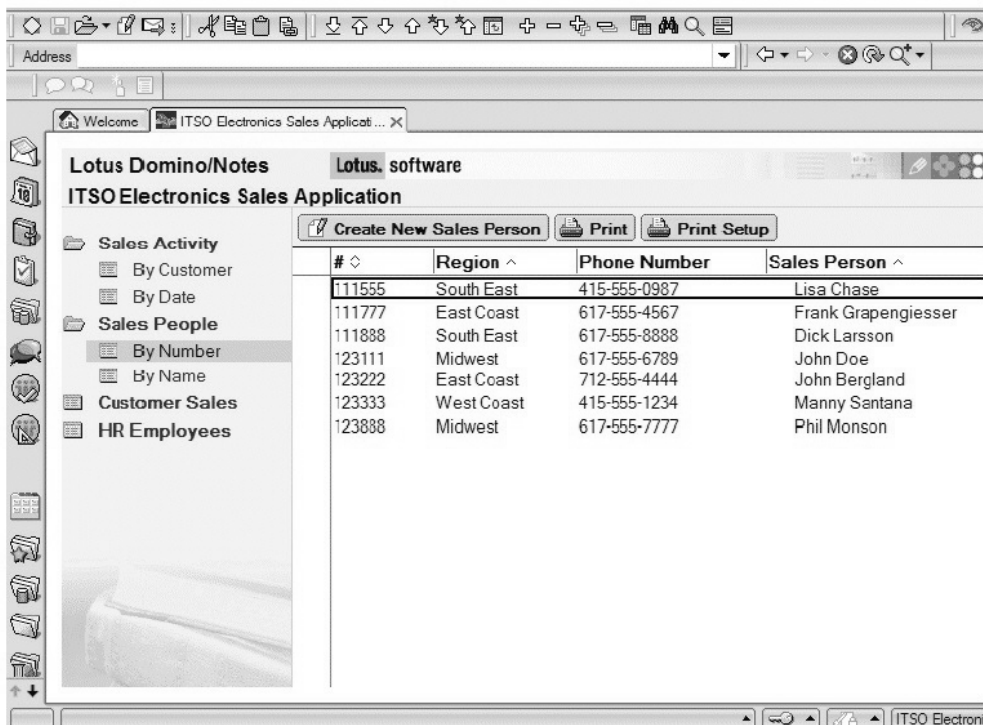


Рис. 4.11. Обновленное представление Sales People By Number со вставленным общим столбцом Sales Person

Для изменения дизайна общего столбца откройте список общих столбцов, щелкнув **Shared Code => Columns** (Общий код => Столбцы), и найдите нужный общий столбец в этом списке. Общий столбец также может быть изменен, для этого необходимо его выделить и щелкнуть **Design => Edit Shared Column** (Разработать => Изменить общий столбец). Предварительно будет предложено сохранить изменения, которые затрагивают описание общего столбца, а значит, автоматически отражаются в представлениях, использующих данный общий столбец, как видно из рис. 4.12. Обновленные представления повторно сохраняются, используя ID текущего пользователя, к тому же в представлениях, использующих опцию **Use Formula Only** (Использовать только формулу), обновляется только формула.

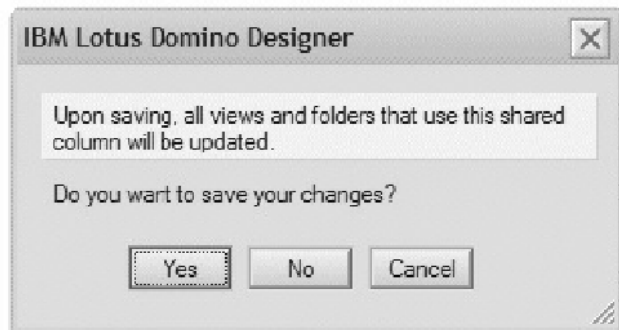


Рис. 4.12. Сохранение общего столбца

Чтобы удалить общие столбцы, выделите нужный столбец в списке общих столбцов и выберите **Edit => Delete** (Изменить => Удалить) или щелкните по нему правой кнопкой мыши и выберите из контекстного меню **Delete** (Удалить) (см. [рис. 4.13](#)). После того как будет удален общий столбец, он станет неотделяемым столбцом во всех представлениях, его содержащих.

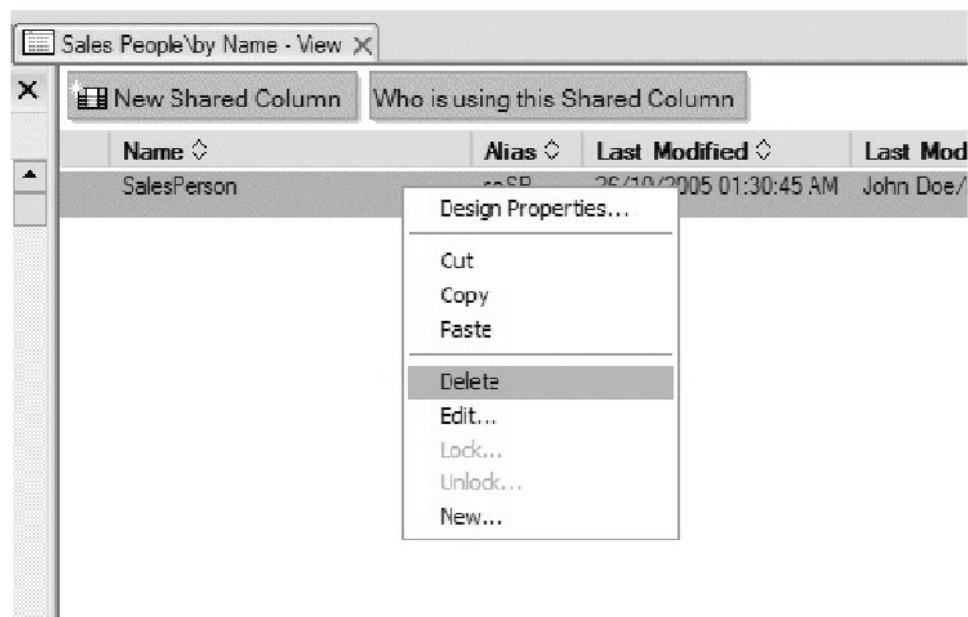


Рис. 4.13. Удаление общего столбца

Щелкнув по командной кнопке Who is using this Shared Column (Кто использует этот общий столбец) в списке Column design, можно увидеть, какие представления и папки в базе данных используют выбранный общий столбец. В приложении ITSO Electronics, если открыть базу данных в Domino Designer, выбрать Shared Code => Columns (Общий код $\Rightarrow$ Столбцы), затем общий столбец SalesPerson из списка и щелкнуть по командной кнопке Who is using this Shared Column (Кто использует этот общий столбец), появится информация, представленная на [рис. 4.14](#).

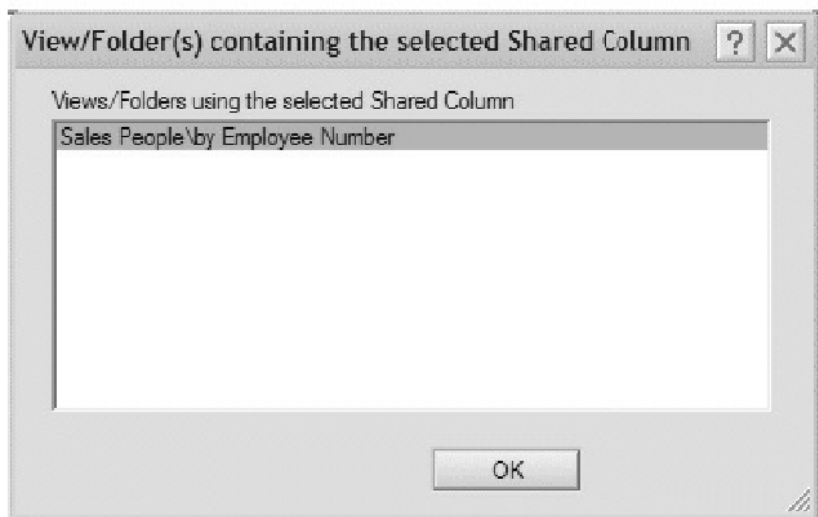


Рис. 4.14. Информационное окно, отображающее все представления, использующие этот столбец

4.4 Расширенные настройки представлений и папок

Свойство столбца, называемое "user definable" (определяемое пользователем), позволяет разработчикам создавать формулу столбцов на основе профильных документов базы данных. В предыдущих версиях программы Lotus Domino Designer использование этой функции было ограничено цветовым кодированием столбца. В Lotus Domino Designer 7 обеспечивается более широкая поддержка профильных документов в столбцах представлений и папок, что позволяет применять в приложениях большее количество параметров времени выполнения.

Хорошим примером в данном случае является почтовый шаблон Lotus Notes 7. В папке Inbox (Входящие) содержится столбец с индикатором письма и скрытый столбец, управляющий цветовым кодированием. Содержимое и режим работы обоих столбцов определяются параметрами почты пользователя, которые пользователь устанавливает самостоятельно (см. [рис. 4.15](#)).

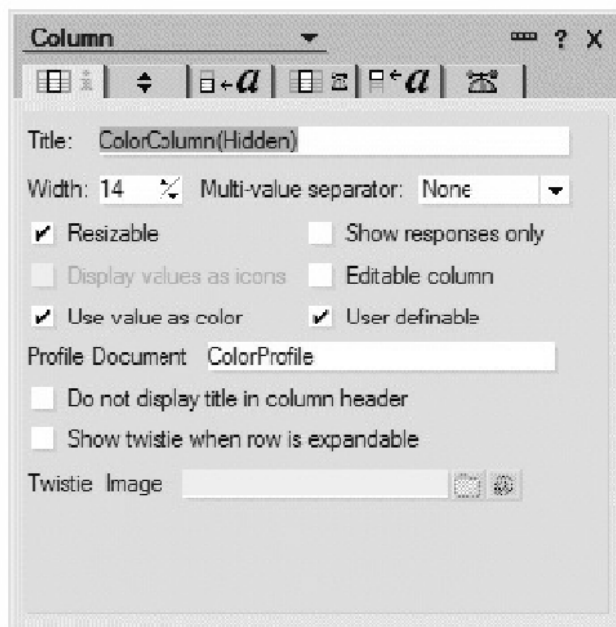


Рис. 4.15. Свойство столбца "User-definable"

Мы хотим улучшить приложение ITSO Electronics с помощью добавления столбца, показывающего регион, выбранный в этом месяце, а также указывающего на то, относится ли документ к текущему пользователю. Иконка в виде звезды отображается в том случае, если данный регион выбран в текущем месяце. Синяя стрелка указывает на то, что данный документ относится к текущему пользователю. В противном случае в этом столбце ничего не отображается. Перед добавлением столбца профиля пользователя мы предполагаем, что приложение содержит профильный документ, который называется SalesProfile. Для столбца профиля пользователя может быть использован только один "общий" профильный документ; следовательно, при создании профильного документа дополнительный

параметр `uniquekey` не используется. За более подробной информацией о профильных документах обратитесь к статье IBM developerWorks  "The hidden power of Profile documents" по адресу: ссылка: http://www.ibm.com/developerworks/lotus/library/ls-Profile_documents/

Для того чтобы добавить новый столбец профиля пользователя в представление "Sales Person By Name" приложения ITSO Electronics, сделайте следующее:

1. Откройте представление в Domino Designer и вставьте новый столбец, щелкнув `Create => Insert new column` (Создать $\Rightarrow$ Вставить новый столбец).
2. Откройте окно свойств `Column` и убедитесь в том, что приведенные ниже значения определены (рис. 4.16):
 - Активируйте опцию `Display values as icon` (Отображать значения в виде иконок), так как столбец профиля пользователя будет отображать иконку, которая соответствует региону, выбранному в текущем месяце, или индикатор, показывающий отношение к текущему пользователю.
 - Активируйте опцию `User definable` (Определяемая пользователем).
 - Определите существующий профильный документ в базе данных `SalesProfile` в поле `Profile Document`.
 - Используйте `@UserName` в качестве формулы значения столбца (`Column Value`).

Замечание Формула значения столбца должна быть непостоянной, такой как `@Username`, `@Created` или `@Random`. В качестве формулы значения столбца можно использовать любую непостоянную формулу, так как ее значение не вычисляется. Если использована постоянная формула, такая как `" "`, или специальная текстовая функция, такая как `@DocNumber`, профильный столбец пользователя не будет отображаться корректно, и для каждого документа в представлении появится сообщение `Replication or Save Conflict` (Конфликт репликации или конфликт сохранения).

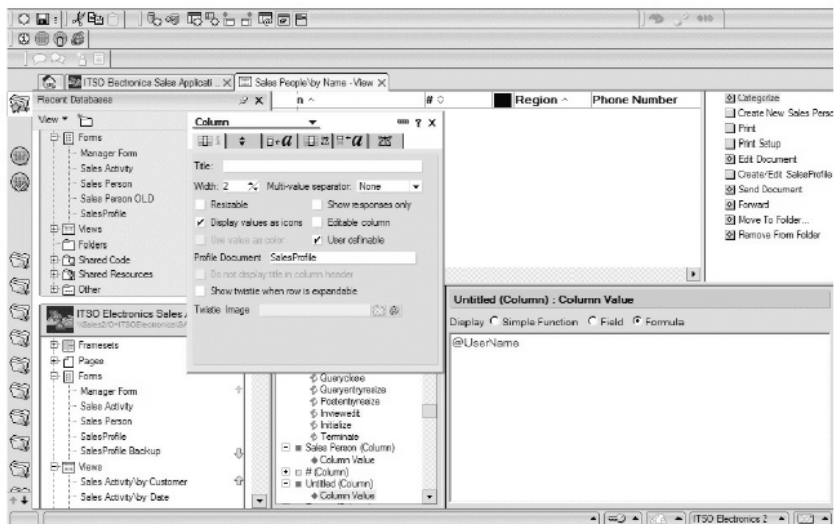


Рис. 4.16. Определение столбца профиля пользователя

- В закладке **Advanced** (см. [рис. 4.17](#)), в поле **Name**, содержится название поля, находящегося в этом документе. В нашем случае это поле `$ToRegion`. В поле профиля документа хранится формула, вычисляемая при отображении строки представления. Другими словами, поле **value** поля `$ToRegion` представляет собой закодированную формулу, выходное значение которой отображается в столбце. Формула **Value**, принадлежащая к `$ToRegion`, представляет собой формулу, выходное значение которой и есть упомянутая формула.

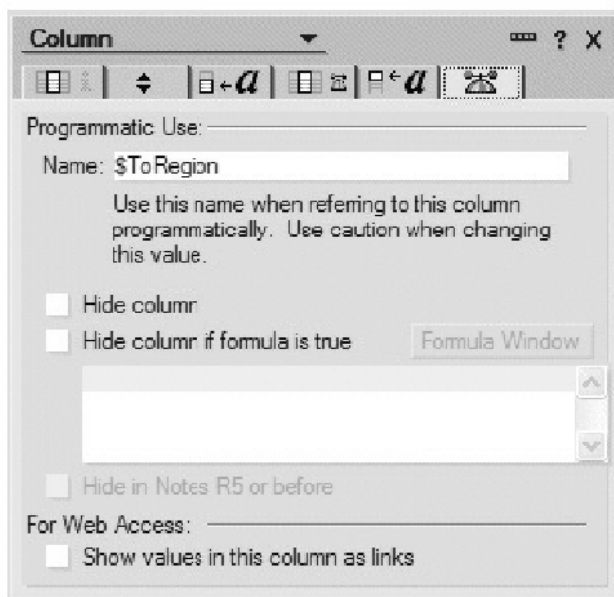


Рис. 4.17. Закладка Advanced в окне свойств Column

Конечный пользователь, редактирующий профильный документ, в большинстве случаев не сможет написать эту формулу. Вычисляемое поле, содержащее формулу, а в нашем случае это `$ToRegion`, обычно вычисляется для конечного пользователя, что необходимо для того, чтобы увидеть, какой должна быть формула. Вычисляемая формула поля должна сгенерировать формулу, которая хранится в поле. На [рис. 4.18](#) представлен дизайн формы `SalesProfile` и показано значение поля в поле `$ToRegion`, которое генерирует значение формулы.

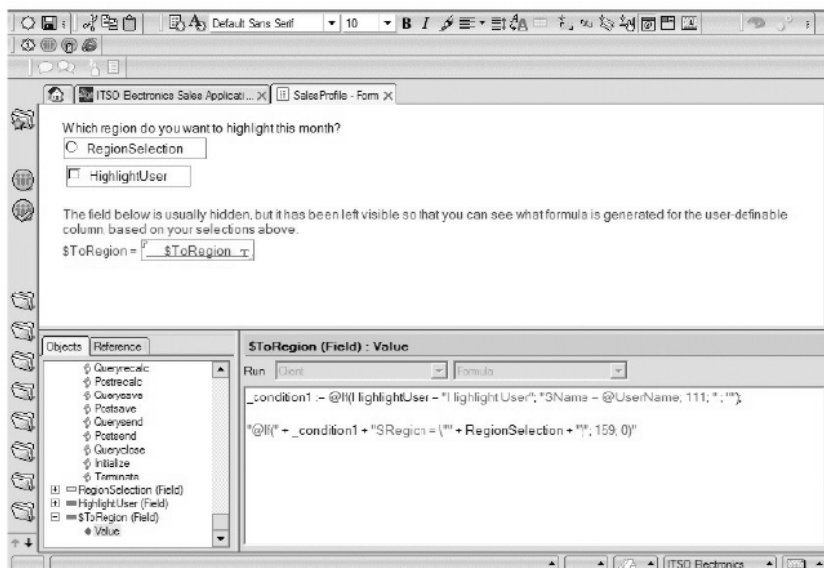


Рис. 4.18. Форма SalesProfile в Domino Designer

3. На рис. 4.19 представлен профильный документ SalesProfile. Введите значения в поля RegionSelection и HighlightUser. Формула поля \$ToRegion вычисляет новую формулу, которая будет использована для столбца, определяемого пользователем. Например, если выбран регион South East и отмечена галочкой опция Highlight User, значение формулы \$ToRegion будет следующим:

@If(SName = @Username; 111; SRegion = "South East"; 159; 0)

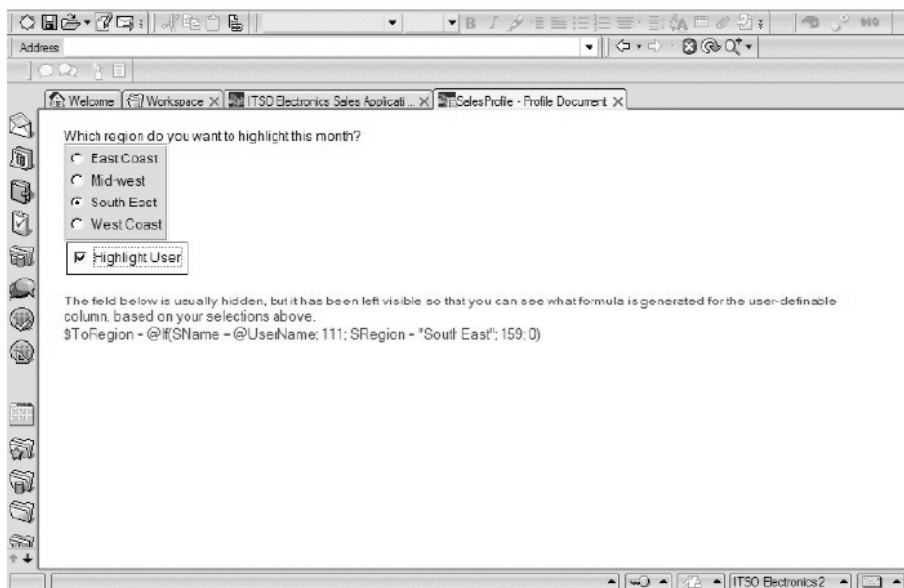


Рис. 4.19. Профильный документ SalesProfile

Опция, выбранная пользователем, будет доступна всегда, а не только когда выбрано *Use value as color* (Использовать цвет в качестве значения). Таким образом, становится возможным использование столбца профиля документа в качестве иконки, а также других типов столбцов. Если профильный документ определен для множества столбцов в границах одного представления, необходимо, чтобы для всех столбцов был использован один и тот же профильный документ. В противном случае при сохранении представления появится сообщение об ошибке (см. [рис. 4.20](#)).

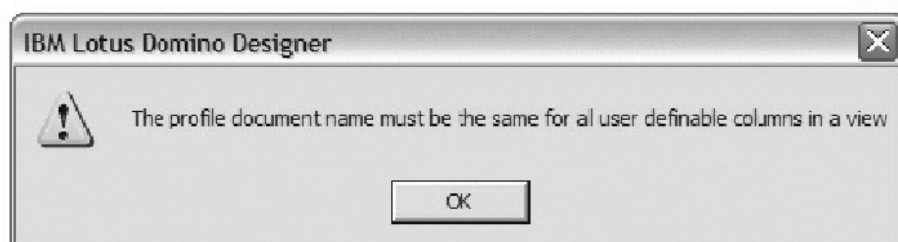


Рис. 4.20. Сообщение об ошибке, касающееся столбца профиля пользователя

После того как представление будет сохранено, столбец профиля пользователя появится в представлении Sales People By Name (рис. 4.21).

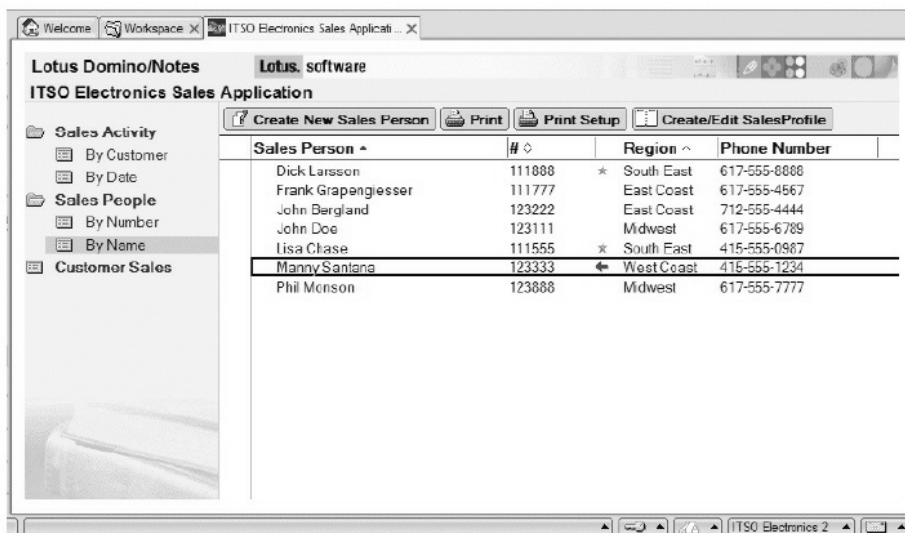


Рис. 4.21. Столбец профиля пользователя, отображающий иконки в представлении

Замечание Будьте осторожны при использовании кавычек и обратных косых черт в вычисляемой формуле поля. Из-за того что множество макроформул содержат кавычки, а некоторые содержат обратные косые черты, вычисляемая формула значения поля профиля часто должна содержать кавычки (или обратные косые черты) внутри кавычек, что приведет к путанице. Поэтому следует делать вычисляемые поля видимыми только на время разработки, как мы и сделали в данном случае, что позволяет редактировать поля, видеть результирующую формулу и определять синтаксически правильные элементы. Также следует помнить о том, что когда вы вставляете данные из других полей в вашу формулу, в них могут содержаться кавычки или обратные косые черты. Поэтому вам придется использовать `@ReplaceSubstring(название поля; "\"": "\""; "\"\\\""; "\"\\\\\"")` для преобразования этих символов в правильные строковые литералы.

4.5 Дополнения к языку формул

Язык формул Lotus Notes обновлен в Domino Designer 7. Новые @-функции внедрены как для поддержки интеграции IBM DB2 и IBM Workplace Client Technology™, так и для поддержки средств безопасности политик и блокирования ECL. Также была добавлена новая @-команда, предназначенная для обнаружения всех папок, в которых располагается выбранный документ.

К новым командам относятся следующие:

- @AdminECLIsLocked: новая функция, которая проверяет текущий статус Administration ECL в адресной книге.
- @DB2Schema: функция, которая по заданному в виде текстовой строки названию базы данных возвращает текстовую строку, содержащую схему DB2 заданной базы данных, если это база данных db2nsf, или же пустую строку, если это база данных не db2nsf. Эта функция полезна в формуле SQL Query в Query View, (см. п. 2.3, "Query View") где вы должны определить формулу, генерирующую SQL-запрос, который возвращает записи, необходимые вам. SQL-запрос должен использовать полное точное табличное имя, включающее название схемы, и желательно использовать именно функцию @DB2Schema, а не название жестко закодированной схемы, чтобы дизайн можно было переносить на другие серверы и приложения.
- @IsDB2: новая функция, которая по заданному серверу и имени файла или серверу и replica-копии ID базы данных сообщает о том, включена ли в данной базе данных поддержка DB2 или нет.
- @IsEmbeddedInsideWCT: новая функция, которая определяет, встроена ли какая-либо часть текущей сессии Notes в клиент Workplace.
- @PolicyIsFieldLocked: новая функция, которая определяет, когда поле заблокировано административной политикой и не может быть изменено.
- @Command([DiscoverFolders]): команда, отображающая окно Folders containing current document (Папки содержащие текущий документ).

Замечание функция @IfError вышла из употребления в Lotus Domino Designer 7.0, однако она полностью совместима с предыдущими версиями.

Для получения дополнительной информации об этих новых функциях и командах см. Lotus Domino Designer 7 Help по адресу: ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf.

4.6 Дополнения к классам LotusScript, COM и OLE

К языку LotusScript были добавлены улучшенные возможности:

- поддержка новых средств профилирования кода, обсуждением которых мы займемся в п. 5.1, "Профилирование кода в приложении ITSO Electronics";
- новые возможности работы с коллекциями документов;
- расширенная поддержка XML;
- необходимая поддержка для интеграции с технологией *Workplace Client*. Также следует отметить одно новое свойство Lotus Domino Designer 7, добавленное к LotusScript. Это свойство класса NotesUIDocument называется ModifiedSinceSaved и выявляет изменения, произведенные в документе в режиме редактирования с момента последнего сохранения.

Для получения дополнительной информации об обновлениях в языке LotusScript в Lotus Domino Designer 7 перейдите к Lotus Domino Designer 7 Help по адресу: ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf

4.7 Дополнения к классам Java и CORBA

Программное обеспечение Lotus Domino Designer 7 поддерживает новые версии Java, *Java Virtual Machine (JVM™)* 1.4.2 и новую версию программы синтаксического анализатора расширяемого языка разметки гипертекста (XML). С помощью отладки Java можно настраивать агенты Java, Web-сервисы и библиотеки скриптов для осуществления удаленной отладки с помощью клиента Lotus Notes.

К Lotus Domino Designer 7 добавлены следующие средства и функции:

- документация к ядру SunTM Java для классов Java и CORBA;
- поддержка платформы Sun JavaTM 2, Technology Edition, V1.4.2;
- новый метод approveHostedOrgStorageDeletion класса AdministrationProcess;
- новый метод getModifiedDocuments класса Database;
- новое свойство UntilTime класса DocumentCollection;
- поддержка Java Platform Debugger Architecture (JPDA) (Архитектура отладчика платформы Java).

Для получения дополнительной информации о поддержке Java и CORBA в Lotus Domino Designer 7 перейдите к Lotus Domino Designer 7 Help по адресу: ссылка:
http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf

Диагностирование и отладка приложения ITSO Electronics

Лекция посвящена отладке приложений: профилирование кода, отладке LotusScript и Java

В Lotus Domino Designer 7 появились новые улучшенные средства диагностирования и отладки. Это профилирование кода, продвинутая отладка LotusScript и *удаленная отладка* Java. Благодаря этим нововведениям и дополнениям можно улучшить производительность приложений Lotus Domino. Взяв в качестве примера приложение ITSO Electronics, мы займемся исследованием некоторых компонентов из инструментария Lotus Domino Designer. Среди них:

- профилирование кода: оптимизация производительности в приложении ITSO Electronics;
- отладка LotusScript: применение улучшенных инструментов отладки LotusScript в приложении ITSO Electronics;
- отладка Java: *удаленная отладка* приложения ITSO Electronics.

5.1 Профилирование кода в приложении ITSO Electronics

В Lotus Domino Designer 7 включен мощный оптимизатор кода, Profiler – инструмент, профилирующий код и позволяющий разработчику приложения измерять время, требуемое для выполнения методов в агентах, Web-сервисах и библиотеках скриптов в приложениях Lotus Notes и Domino. Измерение производительности методов в приложениях Lotus Domino позволяет разработчикам определить недостатки кода и сфокусироваться на оптимизации тех его фрагментов, выполнение которых отнимает больше всего времени, что отражается на производительности. В этом разделе мы рассмотрим, как с помощью профилирования кода достичь максимальной производительности и эффективности приложения ITSO Electronics.

Profiler исследует и оценивает методы как в Java, так и в LotusScript, включая операции над объектами Lotus Domino, такими как база данных Lotus Domino. Можно включить поддержку профилирования в агенте

или в Web-сервисе, для чего следует открыть закладку Security окна свойств и активировать опцию Profile this agent (Профилировать этот агент) или Profile this web service (Профилировать этот Web-сервис). Профилирование происходит каждый раз, когда выполняется агент или Web-сервис, в котором активировано профилирование. Профилирование агента происходит и на клиенте Lotus Notes, и на сервере Lotus Domino. После активирования опции профилирования на второй закладке окна свойств Agent (см. [рис. 5.1](#)) профилирование агента будет производиться, начиная со следующего запуска агента.

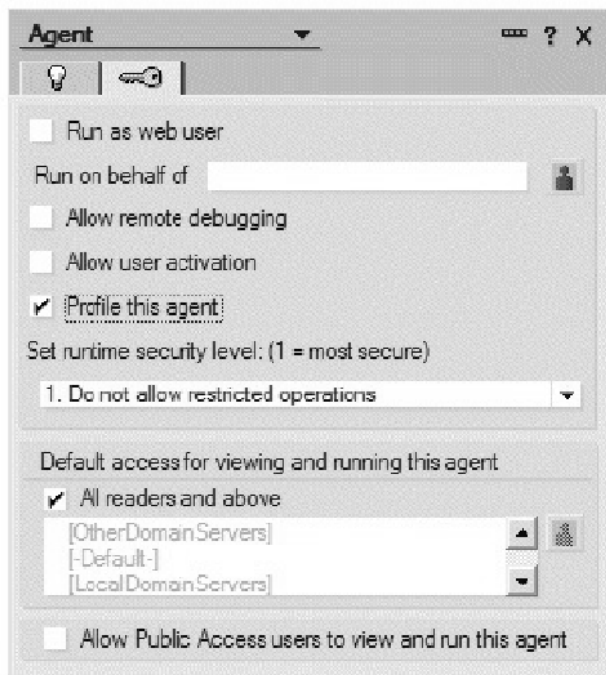


Рис. 5.1. Окно свойств Agent

Агент может быть профилирован вне зависимости от того, запущен он как агент, работающий по расписанию, Web-агент или вручную через меню Action. Профилирующая информация хранится в профильном документе в базе данных, связанной с этим агентом. Для просмотра профилирующей информации выберите профилируемый агент, после чего щелкните Agent => View Profile Results (Агент => Просмотреть результаты профилирования). В верхней части каждого

профильного документа записано название агента и время профилирования. Общее затраченное время равно общему количеству времени, в течение которого работал агент. За ним следует полное измеренное время, которое обычно несколько меньше из-за того, что значения времени округляются в меньшую сторону. Например, значения меньше одной миллисекунды в профильном документе округляются до нуля. В профилирующей таблице содержится строка для каждого вызванного метода Domino Objects, а также 5 столбцов:

- **Class (Класс):** название класса Domino Objects, используя стандартные имена, такие как Session, Database, Document.
- **Method (Метод):** названия метода или свойства Domino Objects, записанные с помощью стандартных имен, таких как CurrentDatabase, AppendItemValue и Save.
- **Operation (Операция):** для свойств, тип операции, либо Get, либо Set.
- **Calls (Вызовы):** количество вызовов метода или свойства.
- **Time (Длительность):** количество времени, затраченного на выполнение вызова в миллисекундах. Символ "<" означает, что времени для вычисления было недостаточно.

Информация в профилирующей таблице упорядочена по убыванию, то есть сначала показаны методы, на выполнение которых ушло больше всего времени. Из профилирующей таблицы этого примера, представленной на [рис. 5.2](#), видно, что метод UpdateFTIndex оказался той частью агента, которая отнимает максимальное количество времени.

ProfileExample Profile

08/25/2005 01:39:53 PM EDT
 Elapsed time: 41059 msec
 Methods profiled: 6
 Total measured time: 37023 msec

Class	Method	Operation	Calls	Time
Database	UpdateFTIndex		27	36113
Database	Open		27	910
Database	Title	Get	27	0
Database	FileName	Get	27	0
DbDirectory	GetNextDatabase		26	0
DbDirectory	GetFirstDatabase		1	0

Рис. 5.2. Результаты профилирования кода

Профилирование кода агентов в реальных приложениях обеспечивает разработчиков полезной информацией, позволяя им определять слабые места в своих программах. Используя эту информацию, разработчик модифицирует логику агента, что экономит время и увеличивает производительность.

5.2 Отладка LotusScript в приложении ITSO Electronics

В Lotus Domino Designer 7 усилены возможности отладки LotusScript за счет большей наглядности средств отладки. В дополнение к запуску отладки через системное меню разработчики приложений теперь могут начинать и прекращать отладку с помощью иконки. Сообщение, указывающее на начало или прекращение отладки LotusScript, располагается на панели состояния, а не в окне сообщений. В этом разделе мы продемонстрируем новые средства диагностики на примере приложения ITSO Electronics.

Для отладки LotusScript-приложения ITSO Electronics с использованием новых возможностей и средств отладки выполните следующие действия:

1. Чтобы включить отладчик, щелкните по кнопке `Debug LotusScript` (Отладить LotusScript), расположенной на панели инструментов. В строке состояния появится сообщение `"LotusScript debugging started"` (рис. 5.3).

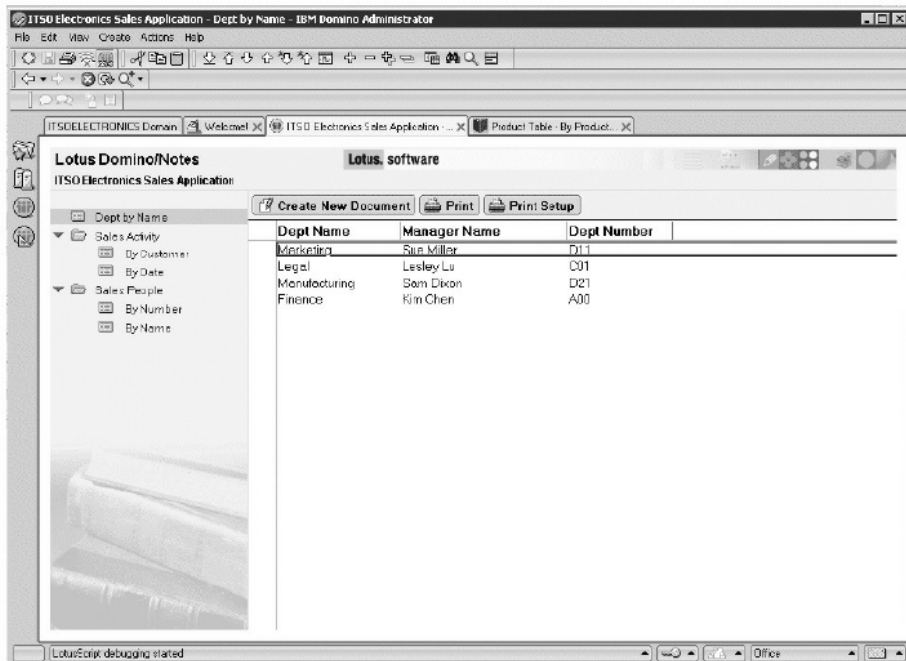


Рис. 5.3. Запуск отладчика LotusScript

2. Чтобы отключить отладчик, снова щелкните по иконке Debug LotusScript. В строке состояния появится сообщение "LotusScript debugging terminated" (рис. 5.4).

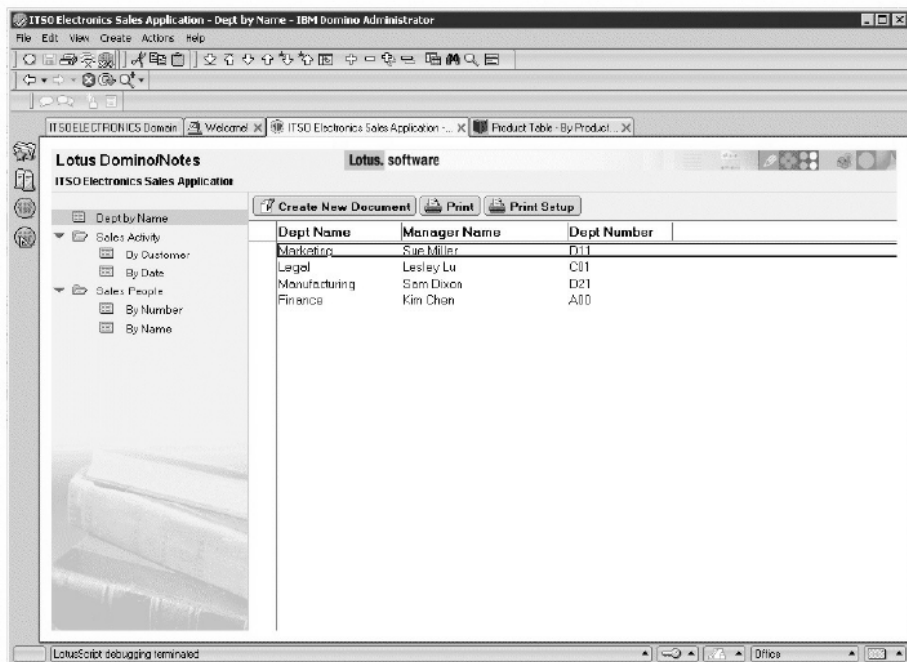


Рис. 5.4. Остановка отладчика LotusScript

5.3 Отладка Java в приложении ITSO Electronics

Пожалуй, самое интересное дополнение к инструментарию отладчика в Lotus Domino Designer 7 – это удаленная отладка Java. Удаленная отладка Java позволяет разработчику приложения в среде Lotus Domino отладить агенты Java, предварительные просмотры в Web и библиотеки скриптов, работающие под управлением клиента Lotus Notes в *Java Virtual Machine (JVM)* с отладчиком, поддерживающим *Java Platform Debugger Architecture (JPDA)*, таким как Rational Application Developer, основанном на технологии Eclipse. И хотя вы не можете отладить код Java, работающий на сервере Lotus Domino, JPDA предлагает богатый набор функций, когда дело касается локальной отладки кода Java. В этом разделе мы продемонстрируем удаленную отладку Java в приложении ITSO Electronics, используя Rational Application Developer.

5.3.1 Включение отладки Java в клиенте Lotus Notes

Клиент Lotus Notes поддерживает отладку Java в представленных ниже контекстах. В каждом контексте есть свой собственный JVM. Код Java из библиотеки скриптов выполняется в контексте вызывающего кода. Отладку в каждом контексте может производить одновременно лишь один пользователь.

- **Foreground (Передний план):** код Java, который интерактивно выполняется в клиенте Lotus Notes, например агент, запускаемый из меню `Actions`.
- **Background (Задний план):** код Java, который выполняется в клиенте Lotus Notes под управлением загрузчика заданий, например локальный агент, работающий по расписанию.
- **Web preview (Предварительный просмотр Web):** код Java, просматриваемый в браузере через Lotus Domino Designer, например апплет на форме.

Для включения отладки Java в приложении ITSO Electronics из клиента Lotus Notes выполните следующее:

1. Выберите `File => Tools => Java Debugging Preferences` (Файл $\Rightarrow$ Сервис $\Rightarrow$ Параметры отладки Java). Откроется окно свойств Java Debugging, представленное на рис. 5.5.

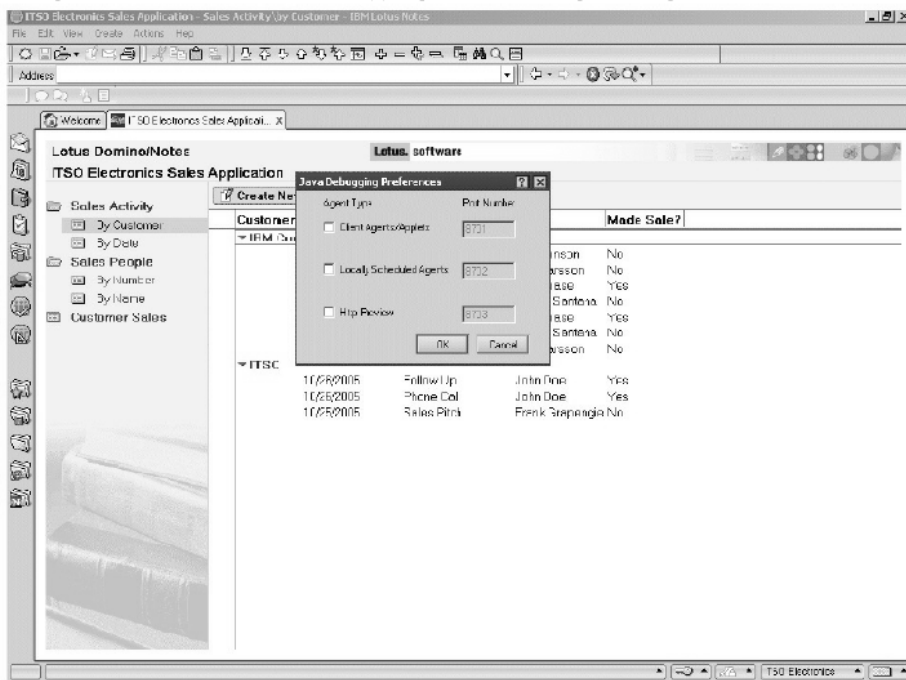


Рис. 5.5. Окно свойств Java Debugging

- Для включения отладки переднего плана (foreground debugging) выберите **Client Agents/Applets** (Агенты/приложения клиента) и определите номер порта, через который будут подключаться отлаживающие компьютеры и Lotus Notes (рис. 5.6). Очистите это поле, если хотите отключить данную функцию.

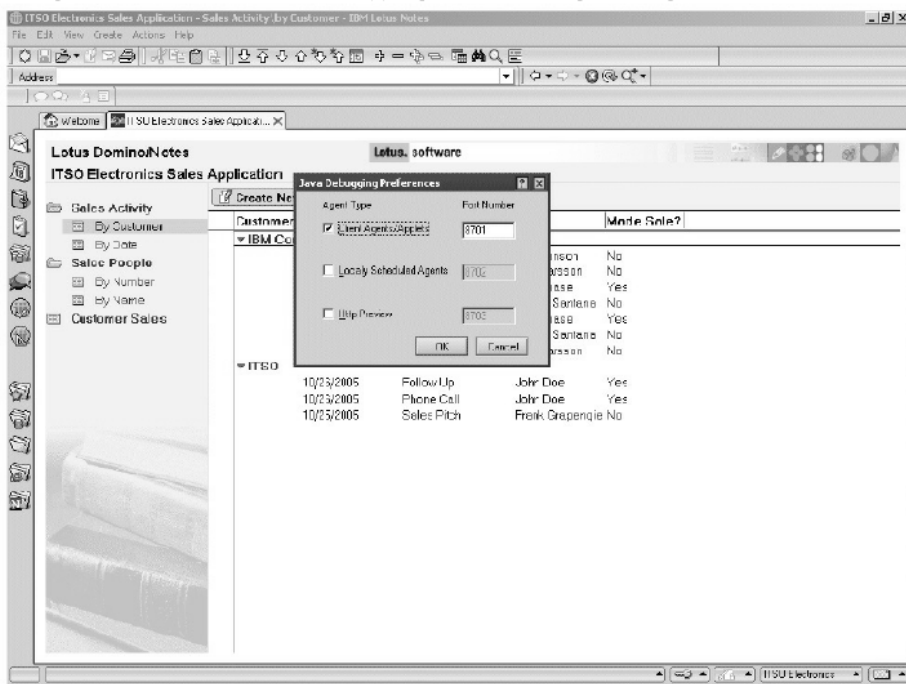


Рис. 5.6. Окно свойств Java Debugging с выбранным типом агента Client Agents/Applets

- Для включения отладки предпросмотра в Web (Web preview debugging), выберите Http Preview (Просмотр в протоколе Http) и определите номер порта, по которому могли бы подключаться компьютеры, на которых запущен клиент Lotus Notes.

Очистите это поле, если не хотите использовать данную функцию. Выберите номер свободного порта. Чтобы узнать, какие порты используются на данном компьютере, работающем под управлением операционной системы Windows, выполните в командной строке команду

```
> netstat -a
```

(для вызова командной строки щелкните Пуск => Выполнить (Start => Run), после чего введите с клавиатуры cmd и нажмите на клавишу Enter). По умолчанию отладка Java

ВЫКЛЮЧЕНА.

Если вы выполняете изменения в параметрах переднего плана (foreground) и заднего плана (background), Lotus Notes необходимо перезапустить. Если вы выполните изменения в параметрах предварительного просмотра Web (Web preview), необходимо будет произвести перезапуск предварительного просмотра.

5.3.2 Включение отладки Java в агенте, Web-сервисе или библиотеке скриптов

Чтобы включить отладку Java, перейдите к закладке Security окна свойств Agent, Web Services или Script Library и выберите Compile Java code with debugging information (Компилировать код Java с предоставлением информации об отладке) (рис. 5.7).



Рис. 5.7. Compile Java code with debugging information

При компилировании кода Java, который будет отлаживаться вне Lotus Domino Designer, компилируйте его с параметром `-g`. После модификации агента или Web-сервиса экспортируйте исходный код в доступный для отладчика файл.

Для получения дополнительной информации о JPDA см.: ссылка: <http://java.sun.com/products/jpda>

5.3.3 Подключение отладчика к JVM

При выполнении приведенных ниже инструкций необходимо использовать Rational Application Developer, основанный на технологии Eclipse. При использовании другого отладчика действия могут

отличаться.

1. Запустите клиент Lotus Notes и откройте приложение ITSO Electronics, содержащее агент или Web-сервис, который необходимо отладить с помощью включенного отладчика Java.
2. Запустите отладчик.
3. Создайте проект Java и переключитесь на перспективу Java (Java perspective), так же как на [рис. 5.8](#). Выберите File => New => Project (Файл => Создать => Проект). Выберите Java в окне Project, введите название проекта и щелкните по кнопке Finish (Завершить). Переключитесь к перспективе Java, либо откройте проект, если он уже существует.

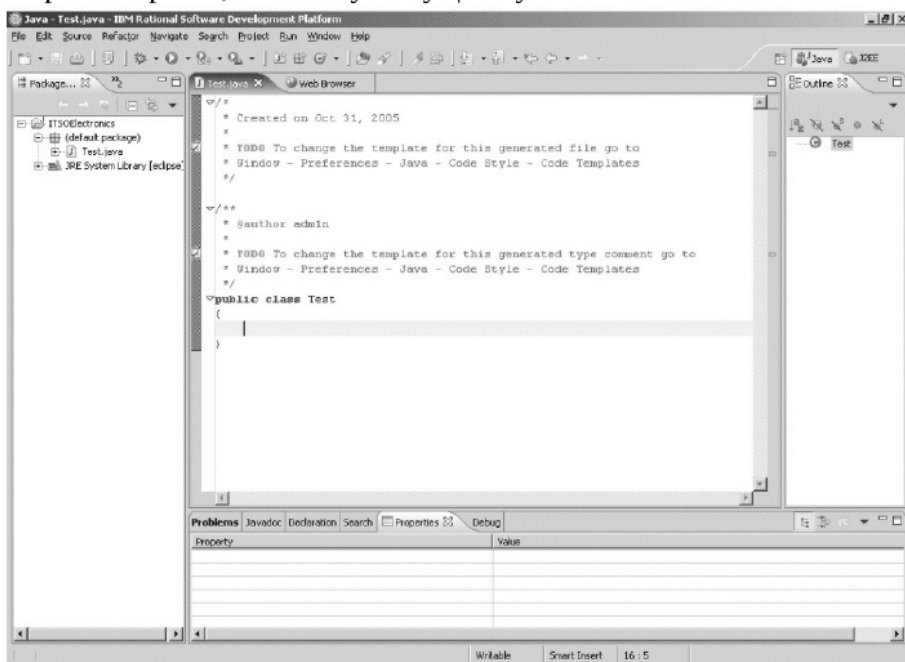


Рис. 5.8. Просмотр перспективы Java

4. Чтобы вставить исходные файлы Java, сделайте щелчок правой кнопкой мыши по папке проекта, находящейся в левой панели, и выберите из контекстного меню Import (Импортировать) (см. [рис. 5.9](#)).

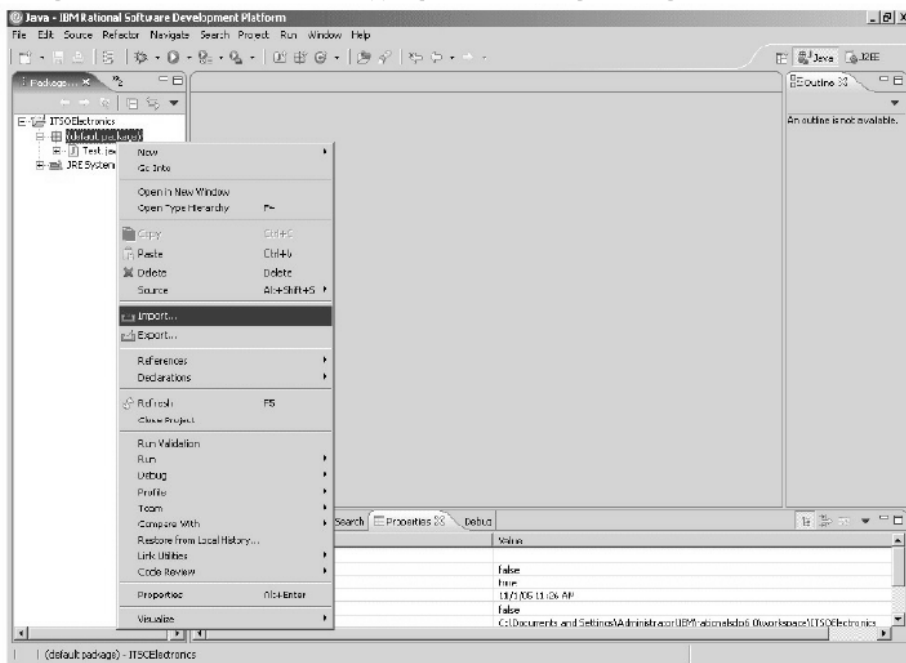


Рис. 5.9. Импорт в перспективу Java

5. Выберите тип файлов, который необходимо импортировать, а затем с помощью функции **Browse** (Обзор) укажите путь к исходным файлам Java. Выберите папку, в которую будут сохранены импортируемые файлы (рис. 5.10).

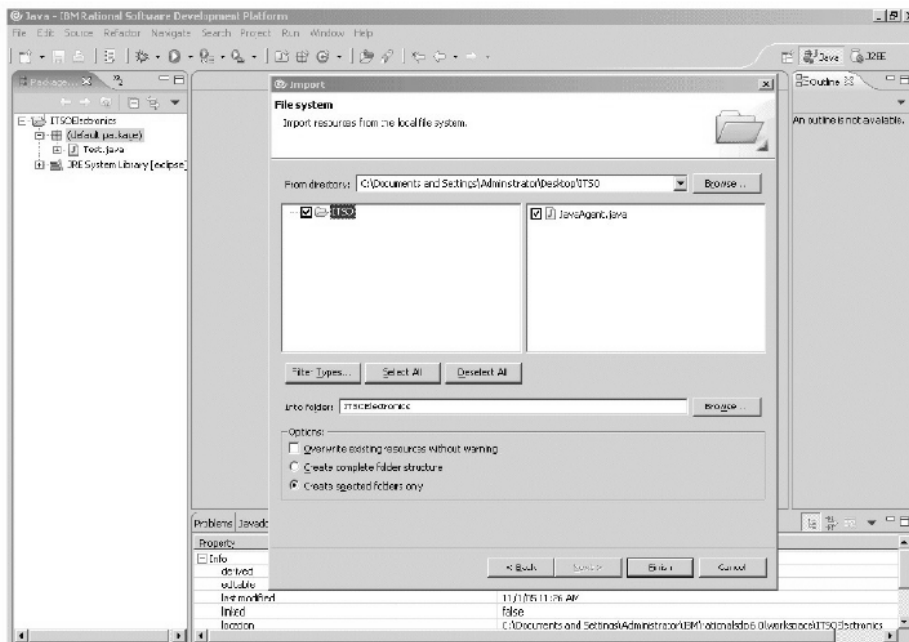


Рис. 5.10. Перспектива Java и окно Important

6. Щелкните по кнопке `Finish` для импорта файлов. Если исходный код Java не импортирован, то вы сможете увидеть поток выполняемых задач и получить доступ к переменным, однако не сможете увидеть исходный код и установить контрольные точки, поскольку отладчику доступен только импортированный исходный код (рис. 5.11).

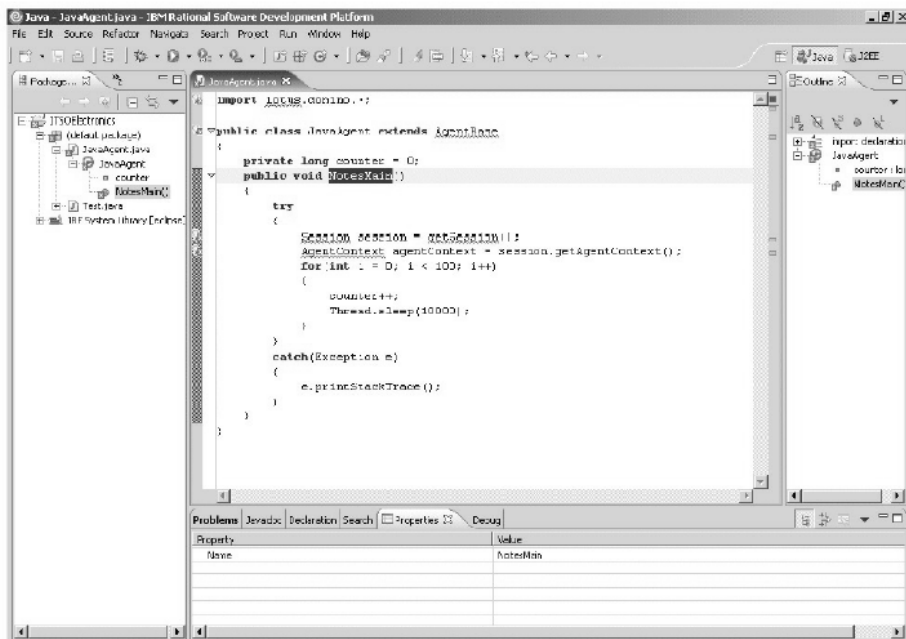


Рис. 5.11. Импортированный исходный код Java

7. В Lotus Notes запустите агент или Web-сервис, который необходимо отладить.
8. В Rational Application Developer выберите Run => Debug (Выполнить => Отладка) и введите название конфигурации отладчика (рис. 5.12). Перейдите к закладке Connect (Подключения). Выберите проект, введите имя хоста или адрес компьютера Lotus Notes и введите номер порта отладки Java в клиенте Lotus Notes. В качестве адреса хоста необходимо использовать 127.0.0.1, если клиенты Lotus Notes и Eclipse находятся на одном компьютере. Щелкните по кнопке Apply (необходимо только для первого раза или если изменена конфигурация), а затем по кнопке Debug, чтобы присоединить отладчик.

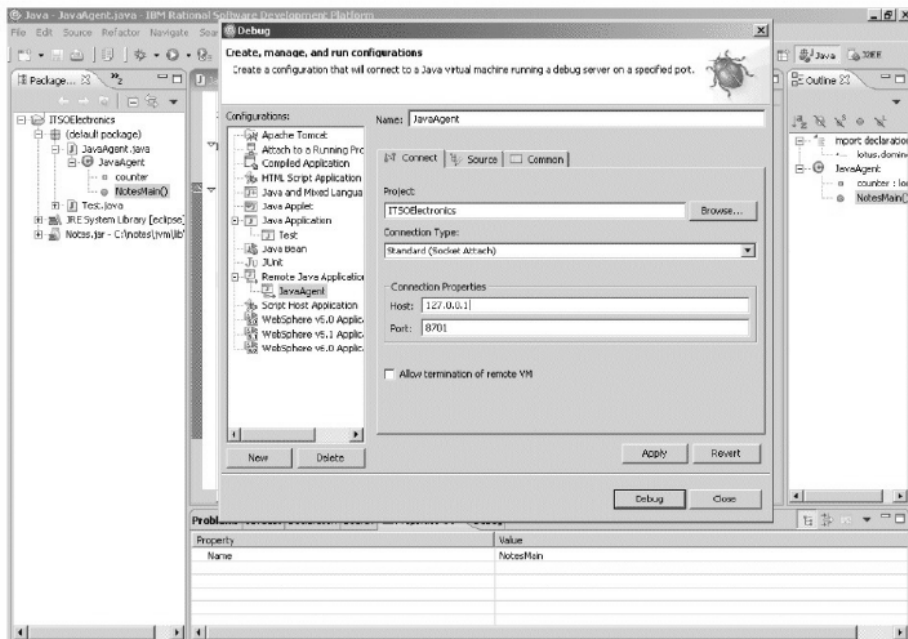


Рис. 5.12. Установка параметров подключения отладчика

После присоединения отладчика вы увидите поток выполняемых задач JVM. Поток агента выглядит следующим образом:

```
Thread(AgentThread: JavaAgent)(Running)
```

Вы можете приостановить поток выполняемых задач, чтобы получить контроль над ним. В этот момент становятся доступными обычные средства отладчика (установка точек прерывания, пошаговая трассировка, продолжение, просмотр значений переменных и т. д.).

Агент Java должен работать достаточно долго, чтобы можно было присоединить отладчик к JVM. Вы можете отложить выполнение в начале процедуры, например вставив цикл с предписаниями ожидания.

Если Lotus Notes и отладчик находятся на разных компьютерах и вы получили сообщение "Failed to connect to remote VM" ("Не удалось подключиться к удаленной VM"), то это, вероятнее всего, свидетельствует о том, что выбранный порт используется другим процессом. Попробуйте определить другой номер порта и

перезапустить Lotus Notes.

Для получения более подробной информации об удаленной отладке Java обратитесь к Lotus Domino Designer 7 Help по адресу: ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf

5.3.4 Рассуждения о безопасности

Отладка Java по сети небезопасна. Вам следует перезапустить Lotus Notes или предварительный просмотр в Web без включенной отладки Java, когда вы не отлаживаете Java-код. Чтобы отключить отладку Java, перейдите к п. 5.3.1, "Включение отладки Java в клиенте Lotus Notes".

Приложение А

В данном руководстве даны ссылки на дополнительный материал, который можно загрузить в Интернете с помощью ресурсов, описанных ниже.

Размещение в Интернете

Электронную версию данного руководства можно загрузить с сервера IBM Redbooks по адресу: <http://www.redbooks.ibm.com/redbooks/REDP4102> ссылка: <http://www.redbooks.ibm.com/>

Как вариант перейдите на Web-сайт IBM Redbooks по адресу: ibm.com/redbooks

Выберите Additional materials и откройте папку, соответствующую номеру руководства – REDP4102.

Использование электронной версии Redbook

Дополнительный материал, сопровождающий данное руководство, содержит файл ITSO Electronics Databases.zip.

Как использовать электронную версию

Создайте на рабочей станции директорию и разархивируйте в нее ZIP-файл с электронной версией руководства. Затем скопируйте базу данных Domino в директорию данных Notes/Domino, например: в c:\notes\data, на вашу локальную рабочую станцию или на ваш сервер Lotus Domino.

Публикации по теме

Публикации, перечисленные в этом разделе, будут особенно полезны для более подробного обсуждения тем, рассмотренных в данном руководстве.

IBM Redbooks

Для получения информации о заказе этих изданий см. п. "Как получить IBM Redbooks". Обратите внимание на то, что часть документов, ссылки на которые приведены ниже, могут быть доступны только в электронном виде.

- Domino Designer 6: A Developer's Handbook, SG24-6854.
- Lotus Domino 6.5.1 and Extended Products Integration Guide, SG24-6357.
- Using the Domino JSP Custom Tags: An Approach to Portalizing Domino Applications, REDP-3902.
- IBM Lotus Domino Application *Portlet*: Configuration and Tips, REDP-3917.
- Security Considerations in Notes and Domino 7, REDP-4104.

Источники в Сети

Для получения дополнительной информации рекомендуется использовать перечисленные ниже Web-сайты и URL.

- IBM developerWorks: Lotus ссылка: <http://www.ibm.com/developerworks/lotus>
- Lotus Domino Administrator 7 Help ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_admin.nsf/
- Web-сайт с документацией по программным продуктам IBM Lotus: ссылка: <http://www.lotus.com/ldd/doc>
- Web-сайт IBM по обучению и сертификации: ссылка: <http://www.ibm.com/software/sw-lotus/services/education.nsf/wdocs/educationhomepage>
- Информационный центр DB2: ссылка: <http://publib.boulder.ibm.com/infocenter/db2help/index.jsp>
- Lotus Domino 7 Designer Help ссылка: http://www.lotus.com/ldd/doc/domino_notes/7.0/help7_designer.nsf
- IBM developerWorks: All about *AutoSave* in Lotus Notes/Domino 7 ссылка: <http://www.ibm.com/developerworks/lotus/library/autosave/>
- IBM developerWorks: Lotus Notes/Domino 7 Web Services ссылка:

<http://www.ibm.com/developerworks/lotus/library/nd7-webservices/>

- IBM developerWorks: Which style of WSDL should I use? ссылка: <http://www.ibm.com/developerworks/webservices/library/ws-whichwsdl/>
- IBM developerWorks: The Hidden Power of Profile Documents ссылка: http://www.ibm.com/developerworks/lotus/library/ls-Profile_documents/
- Developing Applications with IBM Lotus Domino Enabled for DB2, E-learning Edition: Self Paced course, N7D400SP ссылка: <http://www.ibm.com/software/sw-lotus/services/education.nsf/81F04813DC9CE3C7852566DA004C5CB2/>
- Introducing IBM DB2 Concepts course, N7D020 ссылка: <https://education.lotus.com/rw/lewwschd.nsf/5FDC13DD71D11431852565>
- Описание типов данных ссылка: <http://www.w3.org/TR/xmlschema-2>
- Web Service Architecture guide ссылка: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211>
- Apache Axis ссылка: <http://ws.apache.org/axis/>
- Apache SOAP ссылка: <http://ws.apache.org/soap/>
- Java Platform Debugger Architecture (JPDA) ссылка: <http://java.sun.com/products/jpda>

Как получить IBM Redbooks

Искать, просматривать и загружать руководства серий Redbooks, Redpapers, Hints и Tips, черновики публикаций и дополнительные материалы, а также заказывать печатные версии книг серии Redbooks или компакт-диски вы можете на Web-сайте по адресу: ссылка: ibm.com/redbooks

Помощь от IBM

IBM Support and downloads ссылка: ibm.com/support

IBM Global Services ссылка: ibm.com/services

Содержание

Титульная страница	2
Выходные данные	3
Лекция 1. Разработка приложений в среде Lotus Domino 7	4
Лекция 2. Использование интеграции с DB2 для улучшения приложения ITSO Electronics	11
Лекция 3. Улучшение работы приложения фирмы ITSO Electronics с помощью Web-сервисов	73
Лекция 4. Применение новых элементов разработки в приложении ITSO Electronics	128
Лекция 5. Диагностирование и отладка приложения ITSO Electronics	156
Лекция 6. Приложение А	173