bhv

www.bhv.ru
www.bhv.kiev.ua

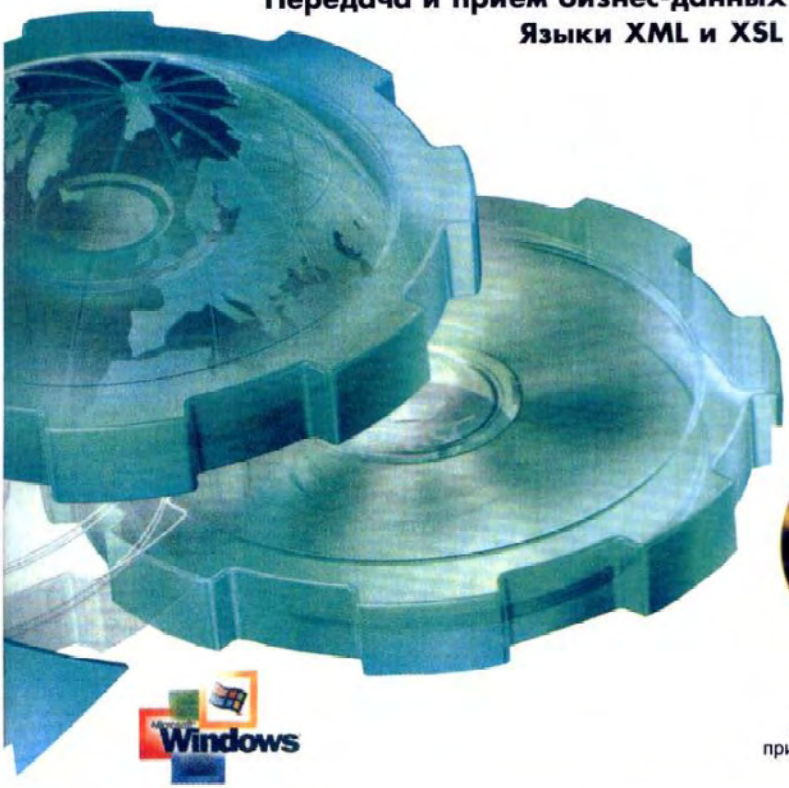
Электронная коммерция

B2B-программирование

ПРАКТИЧЕСКОЕ РУКОВОДСТВО

МАСТЕР

Безопасные конвейеры обработки транзакций
Обработка данных кредитных карт
Передача и прием бизнес-данных
Языки XML и XSL



Windows

CD-ROM содержит образцы сайтов, лабораторные работы, примеры программных кодов

Microsoft Press

Microsoft® Mastering eCommerce Development: Business to Business

Developer Learning Tools

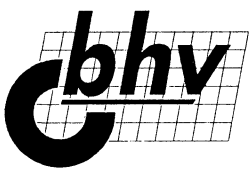
Microsoft®

eCommerce Development: Business to Business

Microsoft Mastering

Электронная коммерция.

B2B-программирование



Санкт-Петербург

Дюссельдорф • Киев • Москва • Санкт-Петербург

УДК 681.3.06

Книга посвящена решению задач электронной коммерции класса business-to-business (B2B) на основе Microsoft Site Server 3.0 Commerce Edition и с использованием программных средств MS Visual Basic, MS Script Debugger, MS Search, MS SQL Server, MS Visual InterDev, MS Internet Explorer. Наряду с фундаментальными понятиями, изложение которых носит обзорный характер, приводится также полный курс лабораторных работ, содержащих много полезных практических примеров: формирование каталогов продукции на сайтах, создание и передача заказов на закупку товаров, обработка данных кредитных карт, регистрация и аутентификация клиентов. Значительное внимание уделено использованию ASP- и XML-технологий, рассматриваемых в ракурсе совершенствования сайтов электронной коммерции в соответствии с требованиями современного бизнеса. Книга снабжена глоссарием и компакт-диском с образцами сайтов, лабораторными работами и демонстрационными примерами программных кодов.

Для широкого круга пользователей

Группа подготовки издания:

Главный редактор	*Екатерина Кондукова*
Зав. редакцией	*Наталья Таркова*
Перевод с английского	*Олега Рябова*
Редактор	*Анна Кузьмина*
Компьютерная верстка	*Ольги Сергиенко*
Корректор	*Зинаида Дмитриева*
Дизайн обложки	*Ангелины Лужиной*
Зав. производством	*Николай Тверских*

Электронная коммерция. B2B-программирование: Пер. с англ. — СПб.: БХВ-Петербург, 2001. — 368 с.: ил.

ISBN 0-7356-1043-6 (англ.)
ISBN 5-94157-057-0

**Лицензия ИД № 02429 от 24.07.00. Подписано в печать 16.02.01.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 29,67.
Тираж 4000 экз. Заказ** 827
"БХВ-Петербург", 198005, Санкт-Петербург, Измайловский пр., 29.

**Гигиеническое заключение на продукцию, товар, № 77.99.1.953.П.950.3.99
от 01.03.1999 г. выдано Департаментом ГСЭН Минздрава России.**

**Отпечатано с готовых диапозитивов
в Академической типографии "Наука" РАН.
199034, Санкт-Петербург, 9-я линия, 12.**

Содержание

Об учебном курсе

Цель этой книги — научить вас тому, как улучшить поисковые возможности вашего сайта, использовать Commerce Interchange Pipeline Manager для связи с бизнес-партнерами, а также применять XML и XSL для управления представлением и передачей данных между бизнес-партнерами.

Для данного курса предполагаются хорошие знания Windows NT 4.0 Security, VBScripting, HTML, DHTML ADO, а также сведений, выводимых программами-мастерами Microsoft Site Server Commerce Edition Wizards.

Содержание курса

В содержание данного курса включены следующие главы.

- **Глава 1. Основные понятия электронной коммерции класса business-to-business**

  Знакомит обучающихся с электронной коммерцией класса business-to-business, характеристиками безопасности сайта этого класса и показывает, как производится регистрация и аутентификация электронных партнеров.

- **Глава 2. Формирование каталогов и поиск**

  Описывает порядок использования Catalog Build Server и Search Server для формирования каталогов и осуществления поиска в них.

- **Глава 3. Создание и передача заказа**

  Демонстрирует порядок создания и передачи заказа на закупку с помощью Commerce Interchange Pipeline (CIP).

- **Глава 4. Получение заказа**

  Демонстрирует обучающимся, как с помощью CIP осуществляется прием заказа на закупку.

- **Глава 5. Создание компонентов конвейера с помощью CIP Manager и работа с ними**

  Раскрывает порядок разработки приложений с помощью CIP Manager.

- **Глава 6. Обзор XML**

  Знакомит с созданием и проверкой XML-документов.

- **Глава 7. Обзор расширяемого языка стилей XSL**

  Рассматривается порядок работы с XML-документами, в которых используются таблицы стилей XSL.

- **Глава 8. COM-объекты в XML**

  Знакомит обучающихся с тем, как создавать COM-объекты в скрипт-языках с помощью XML.

- **Глава 9. Электронный обмен данными EDI**

  Описывается, как осуществлять конвертирование данных из XML в EDI и обратно.

- **Глава 10. Сохранение данных о состоянии сеанса**

  Показано, как сохраняются данные о состоянии сеанса на сайте с помощью Active User Object (AUO) и Membership Directory.

- **Глава 11. Обработка данных кредитных карт**

  Рассматривается порядок обработки данных о кредитных картах и работа с Microsoft Passport на сайте.

- **Глава 12. Настройка сайтов в соответствии с потребностями пользователей**

  Показан порядок настройки сайтов, созданных с помощью Site Builder.

Лабораторные работы

Предлагаемый курс содержит лабораторные работы, которые позволяют обучающимся получить практические навыки работы со средствами и технологиями, представленными в главах. Каждая лабораторная работа состоит из одного и более упражнений, направленных на практическое использование материала главы.

Подготовка лабораторных работ

Для выполнения упражнений и просмотра соответствующих программных кодов вам необходимо инсталлировать файлы лабораторных работ с прилагаемого к данной книге компакт-диска. Более подробная информация об этой процедуре приведена в *приложении B*.

Для выполнения лабораторных работ необходимо установить следующее программное обеспечение:

- Microsoft Windows NT 4.0 Server;
- Microsoft Visual Studio 6.0 Enterprise Edition;

- Windows NT Server 4.0 Service Pack 3;
- Microsoft Internet Explorer 4.01 SP1;
- Microsoft Visual Basic 6.0;
- Microsoft Visual InterDev 6.0 (не обязательно);
- MSDN Library (не обязательно);
- Windows NT 4.0 Option Pack;
- Microsoft FrontPage 98 Server Extensions version 3.0.2.1706;
- Windows NT Server 4.0 Service Pack 4;
- Internet Explorer 5.0;
- Microsoft SQL Server 7.0 Standard Edition;
- Microsoft Data Access 2.1 (MDAC);
- Microsoft Site Server 3.0;
- Microsoft Site Server 3.0 Commerce Edition;
- Site Server 3.0 Service Pack 2;
- Active Directory Services Interface 2.5 (ADSI);
- Commerce Interchange Pipeline Manager;
- Windows NT Server 4.0 Service Pack 5.

Вопросы для повторения

В данном курсе, в конце каждой главы предусмотрено несколько вопросов для повторения пройденного материала. Вы можете использовать их для самопроверки[1].

Содержимое компакт-диска

Содержимое данного CD-ROM[2] необходимо просматривать в HTML-браузере с поддержкой фреймов (frames). Если у вас на компьютере не установлен браузер или же не поддерживаются фреймы, вы можете воспользоваться копией Microsoft Internet Explorer, содержащейся на этом компакт-диске. При этом руководствуйтесь инструкцией по инсталляции, размещенной в файлах ReadMe.txt и ReadMe_rus.txt.

Для просмотра содержимого диска откройте файл default.htm.

---

[1] Ответы на вопросы приведены в конце книги. — *Ред.*

[2] Описание содержимого компакт-диска представлено в *приложении Е*. — *Ред.*

Файлы лабораторных работ

На компакт-диске также размещены файлы, необходимые для выполнения упражнений лабораторных работ, а также соответствующие программные коды.

Примечание

Для инсталляции файлов лабораторных работ требуется 15 Мбайт дискового пространства.

Соглашения, используемые в книге

Ниже приводится разъяснение соглашений, используемых в этой книге.

Пример соглашения	Описание
Меню **File**, **Add Project**	Большинство элементов интерфейса выделено полужирным шрифтом того же размера, что и основной текст
http://www.microsoft.com/	Полужирный шрифт используется также для указания URL
Variable	В синтаксических конструкциях, курсивом выделяются замещаемые элементы, а в тексте — специальные термины
[expressionlist]	В синтаксических конструкциях, элементы, расположенные в квадратных скобках, являются необязательными
`{While | Until}`	В синтаксических конструкциях, фигурные скобки и вертикальная черта обозначают выбор одного элемента из некоторого множества элементов

`Sub HelloButton_Click()`
`  Readout.Text = "Hello, world!"`
`End Sub`
	Этот шрифт используется для отображения программного кода, а также данных, вводимых пользователем

ГЛАВА 1

Основные понятия электронной коммерции класса business-to-business

Обзор

Электронная коммерция класса business-to-business (business-to-business e-commerce) представляет собой систему осуществления электронных транзакций (взаимодействий) между двумя деловыми партнерами (процессами). При таком взаимодействии каждая сторона получает определенные преимущества, в частности, снижение временных затрат на взаимодействие и низкие закупочные цены.

Так как электронная коммерция предполагает обмен конфиденциальной информацией, Web-сайты, ориентированные на работу с системами класса business-to-business[1], должны обеспечивать должный уровень безопасности информации. Следовательно, транзакции должны устанавливаться лишь между доверительными сторонами, именуемыми *электронными партнерами* (e-Partners). Только им предоставляется доступ к охраняемой конфиденциальной информации.

Цели и задачи

После прочтения данной главы вы научитесь:

- понимать суть электронной коммерции класса B2B;
- объяснять сущность цепочки добавления стоимости (value chain) в бизнесе;

[1] Деловые отношения между компаниями (т. е. электронную коммерцию класса business-to-business) принято обозначать аббревиатурой B2B. Поэтому далее будет использоваться такое обозначение данного типа взаимодействия. — *Ред.*

- объяснять необходимость применения систем электронной коммерции класса B2B в условиях современного рынка;
- объяснять некоторые типичные бизнес-сценарии (business scenarios), реализуемые в Web;
- отличать системы электронной коммерции класса B2B от систем класса business-to-consumer;
- формулировать требования, предъявляемые к организациям, вырабатывающим решения в области электронной коммерции;
- объяснять, как обеспечивается безопасность информации бизнес-транзакций (business transactions);
- создавать группу e-Partner на Web-сайте.

Примечание

Коды, приводимые в данной главе, находятся на CD-ROM[1] в файле Practices\Mod01\Mod01Code.txt.

Системы электронной коммерции

В данном разделе обсуждаются:

- понятие цепочки добавления стоимости (Value Chain);
- введение в электронную коммерцию класса B2B;
- достоинства электронной коммерции класса B2B;
- сценарии, применяемые в электронной коммерции класса B2B;
- сравнение систем электронной коммерции классов business-to-business и business-to-consumer.

Типичная организация (предприятие) состоит из нескольких основных и вспомогательных подразделений, которые обеспечивают общую слаженную работу. Например, некоторая сервисная организация может иметь в своем составе одно подразделение, непосредственно обслуживающее клиентов, а также подразделение, которое занимается хранением информации о клиентах и различных данных. В состав сервисной организации могут входить такие подразделения, как отдел кадров, отдел информационных систем, финансовый отдел, которые обеспечивают работу основных подразделений. Все подразделения работают вместе для обеспечения требуемого уровня сервиса клиентов.

Для того чтобы повысить уровень обслуживания своих клиентов, организация должна также тесно взаимодействовать со своими поставщиками.

[1] Если рабочие файлы инсталлированы на жесткий диск, то необходимый код будет размещен в файле <папка_установки>\DemoCode\Mod02\Mod02Code.txt. — *Ред.*

Именно с этой целью многие организации в настоящее время обращаются к электронной коммерции как к средству, обеспечивающему удобное взаимодействие (транзакции) между организациями.

В последующих разделах мы расскажем о достоинствах электронной коммерции класса B2B, о ключевых приложениях, относящихся к этому классу, а также о стандартных сценариях B2B.

- В *разд. "Понятие цепочки добавления стоимости"* излагается последовательность деятельности организации (производства) в направлении повышения уровня обслуживания клиентов.
- В *разд. "Введение в электронную коммерцию класса business-to-business"* содержится информация об основных понятиях электронной коммерции B2B.
- *Раздел "Преимущества электронной коммерции класса business-to-business"* познакомит вас с преимуществами использования систем данного класса.
- В *разд. "Сценарии"* вы узнаете, какие типы организаций и бизнес-сценариев позволяют реализовать преимущества электронной коммерции B2B.
- Наконец, в *разд. "Сравнение систем электронной коммерции классов business-to-business и business-to-consumer"* рассматриваются различия, существующие между этими двумя классами систем.

Понятие цепочки добавления стоимости

Основная деятельность типичной организации заключается в разработке, производстве, маркетинге, распространении и поддержке продукции или услуг. Каждая из этих сфер деятельности привносит прирост стоимости продукта или услуги, предоставляемых клиенту. Цепочка видов деятельности, направленных на удовлетворение нужд заказчика называется *цепочкой добавления (прироста) стоимости* (value chain).

На рис. 1.1 показаны основные виды деятельности, свойственные большинству организаций (предприятий).

Цепочка добавления стоимости

Цепочка добавления (прироста) стоимости указывает пять основных и четыре дополнительных вида деятельности, обуславливающих рост стоимости продукции (сервиса).

Например, в бизнес-сценарии производства основными видами деятельности являются:

- закупка сырья;
- переработка сырья в готовую продукцию;

Рис. 1.1. Виды деятельности организации

- отправка готовой продукции розничным торговцам или дистрибьюторам;
- маркетинг продукции;
- обслуживание (сопровождение) продукции.

В этом бизнес-сценарии к вспомогательным видам деятельности относятся:

- получение различной исходной информации (inputs) для каждого основного вида деятельности;
- развитие технологии;
- управление людскими ресурсами;
- управление инфраструктурой.

Достижение и удовлетворение требований заказчика

Для сохранения конкурентоспособности организация должна правильно оценивать стоимость и производительность каждого вида деятельности в цепочке добавления стоимости и находить пути снижения затрат и повышения эффективности производства. Успех организации (предприятия) зависит не только от того, насколько хорошо работает каждое из его подразделений, но также от того, насколько правильно скоординированы действия этих подразделений.

Для достижения гарантированного качества сервиса компаниям необходимо эффективно управлять основными бизнес-процессами. Такие процессы могут включать в себя:

- исследование, усовершенствование и выпуск качественно новых изделий, осуществляемые быстро и в рамках выделенного бюджета;
- разработку и поддержание на адекватном уровне запасов сырья, полуфабрикатов и готовой продукции;
- деятельность, относящуюся к приему заказов, своевременной отгрузке товаров и получению платежей;
- деятельность, направленную на обеспечение быстрого и исчерпывающего сервиса и ответы на запросы клиентов.

Автоматизация как средство снижения цен и роста эффективности

Для усовершенствования сервиса клиентов и удовлетворения их запросов предприятие должно минимизировать затраты и повышать производительность на каждом звене цепочки добавления стоимости. Предприятие может достичь эту цель путем автоматизации взаимодействия:

- между своими функциональными подразделениями (основными и вспомогательными);
- между предприятием и его поставщиками;
- между предприятием и его клиентами.

Автоматизация этих взаимодействий снижает цены (затраты) и повышает уровень удовлетворения клиентов следующими путями:

- сокращая производственный цикл;
- снижая уровни запасов;
- повышая оперативность отклика на запросы клиентов;
- совершенствуя обмен данными между функциональными подразделениями данного предприятия.

Введение в электронную коммерцию класса business-to-business

Организации различного масштаба быстро продвигаются в направлении электронной коммерции на основе Internet, сетей intranet, extranet, а также так называемых сетей с "добавленной стоимостью" (дополнительными услугами) (value-added networks) для того, чтобы усовершенствовать взаимодействие между фирмами (business-to-business interaction).

Электронная коммерция класса business-to-business

Электронная коммерция класса B2B всецело относится к автоматизации процессов, используемых покупателями и продавцами для ведения бизнеса.

Эти процессы обычно включают в себя онлайновую продажу (online sale) товаров и услуг в Web.

При электронном способе взаимодействия между покупателем и продавцом время отклика значительно снижается, особенно при использовании стандартных коммуникационных форматов. Например, если производитель стандартизирует (нормирует) формат заказа на поставку, то затем может быть также стандартизирован (нормирован) процесс получения и обработки заказа на поставку. Одним из способов стандартизации транзакций в электронной коммерции заключается в использовании *EDI* (Electronic Data Interchange, электронного обмена данными[1]). EDI стандартизирует как обмен внутри организации, так и между организациями, повышая, таким образом, производительность и снижая эксплуатационные расходы.

Пример Web-сайта с электронной коммерцией класса business-to-business

Web-сайт Microsoft Market (MS Market) представляет собой образец сайта электронной коммерции класса B2B, созданный на основе реального внутреннего Web-сайта корпорации, ориентированного на закупки. На этом сайте-образце сотрудники могут заказывать товары для административных нужд и офиса напрямую у различных поставщиков в разных странах, используя Web-браузер, установленный на настольном компьютере. Данный Web-сайт экономит рабочее время сотрудников, обеспечивая непосредственные связи с предпочтительными каталогами поставщиков, проверяя уровни полномочий (signature authority level), выясняя действительность счета и код центра учета (cost-center coding), автоматически поставляя там, где это возможно, информацию для заказов на закупку.

Когда сотрудник инициализирует запрос, MS Market генерирует учетный номер заказа (order-tracking number) для дальнейших ссылок, отправляет извещение менеджеру данного сотрудника по электронной почте, а затем направляет заказ в электронном виде поставщику для исполнения.

Преимущества электронной коммерции класса business-to-business

Электронная коммерция обеспечивает преимущества интеграции бизнес-процессов. Например, отдел закупок приобретает каталог продукции у поставщика и обеспечивает доступ к нему пользователей intranet. Пользовате-

---

[1] В словаре терминов книги Д. Козье "Электронная коммерция": Пер с англ. — М.: Русская редакция, 1999., приводится следующее определение: EDI — электронный обмен деловыми документами (такими, как заказы на покупку, котировки, накладные и счета-фактуры) между компьютерными программами различных компаний в стандартизированной форме. Обычно системы EDI используются при ведении дел с поставщиками. — *Пер.*

ли просматривают каталог и заказывают продукцию. Этот процесс заказа продукции и, соответственно, генерации счетов является автоматизированным. Допустимый объем пользовательского заказа также определен и является частью автоматизированного процесса.

На рис. 1.2 отображены пять основных преимуществ применения электронной коммерции.

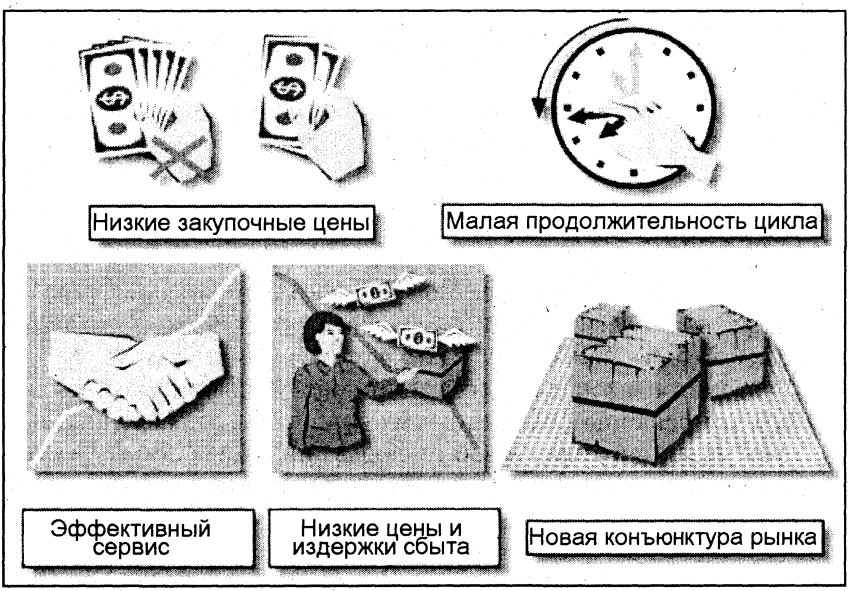


Рис. 1.2. Основные преимущества применения электронной коммерции

До недавнего времени воспользоваться преимуществами электронной коммерции могли только самые крупные компании. Сегодня, однако, с развитием Internet это стало доступным и для малых предприятий. Организации любого масштаба могут взаимодействовать друг с другом на основе электронных технологий не только при помощи Internet, но также посредством intranet, extranet или частных интегрированных сетей (value-added networks).

Вот несколько неоспоримых доводов, свидетельствующих о целесообразности использования электронной коммерции класса B2B.

Низкие закупочные цены

Предприятия снижают закупочные цены (procurement costs) развивая отношения с основными поставщиками с целью получения прибыльных процентов скидки (volume discounts). Internet теперь позволяет предприятиям взаимодействовать с новыми поставщиками, в том числе среднего и малого масштаба, которые часто предоставляют материалы по низким ценам. Inter-

net также уменьшает производственные издержки (processing costs) и открывает новую конъюнктуру рынка за счет покупателей, запрашивающих цены в онлайновом режиме.

Малая продолжительность цикла

Создавая электронные связи со своими поставщиками и клиентами, предприятия могут передавать и получать заказы на поставку (purchase orders), счета, а также уведомления об отгрузке (shipping notifications) в более короткие сроки.

Электронная связь с поставщиками, производителями и клиентами позволяет предприятиям поддерживать минимальные оборотные фонды (запасы), существенно снижая относительные цены (related costs).

Эффективный сервис

Предприятия, которые занимаются электронной коммерцией, предоставляют в онлайновом режиме информацию о продукции, технической поддержке и состоянии заказа (order status). Это позволяет сберечь денежные средства путем высвобождения персонала от работы по обслуживанию клиентов и использования его для решения более сложных вопросов и управлению связями с заказчиками. Наличие онлайновой информации способствует удовлетворению потребностей тех клиентов, которые ценят ее доступность (ease of access).

Низкие цены и издержки сбыта

Связь предприятий посредством Internet может привлечь новых клиентов без дополнительных затрат. Это обусловлено тем, что функция сбыта (sales function) предприятия возлагается на сервер, а не на продавцов и торговые учреждения (store locations). Возможности предприятия ограничиваются лишь производительностью серверов, используемых для обработки запросов и заказов.

Новая конъюнктура рынка

Internet функционирует круглосуточно и по всему миру, поэтому те предприятия, которые подключены к Internet в онлайновом режиме, имеют доступ к таким новым рынкам сбыта, работа с которыми не была бы эффективной в случае использования традиционных торговых агентов и развертывания рекламной кампании.

Имея онлайновый доступ и создавая заказные сервисы для рынка предприятия, поставщики могут осваивать рентабельные новые рынки как в локальном, так и в глобальном масштабе.

Примечание

Указанные преимущества действительны лишь в том случае, если в цепочке добавления (прироста) стоимости имеются все звенья и созданы условия для интеграции бизнес-процесса.

Сценарии

В настоящее время можно выделить несколько типов электронной коммерции.

На рис. 1.3 показаны три главных сценария электронной коммерции класса business-to-business.

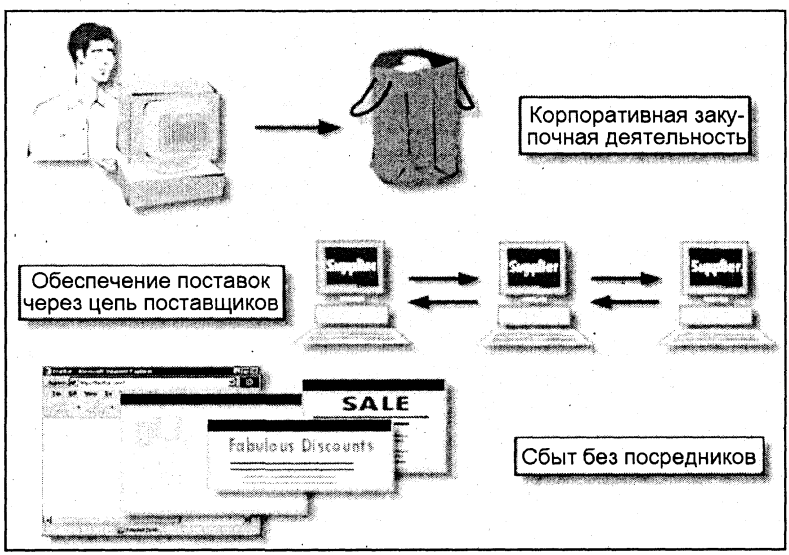


Рис. 1.3. Главные сценарии электронной коммерции класса business-to-business

Далее приводятся три сценария, которые являются наиболее распространенными.

Сценарий 1: корпоративная закупочная деятельность

Корпоративная закупочная деятельность представляет собой приобретение материалов для использования служащими компании. Эти материалы не применяются в процессе производства. Корпоративная закупочная деятельность обычно предполагает обмен бумажными формами документов и требует участия менеджеров по закупкам. Автоматизация этого процесса снижает рабочую нагрузку управленческого персонала, а также связанные с

этим процессом затраты и работу по подсчету (overall headcount), неизбежную при работе с бумажными документами.

Например, MS Market позволяет любому сотруднику "посетить" так называемый "торговый пассаж" (supplies mall) или зону электронной торговли, созданную на основе Web и имеющую централизованное управление. MS Market является неким виртуальным предприятием (virtual enterprise), в котором участвуют Microsoft и ряд высокоэффективных компаний-поставщиков.

Сценарий 2: обеспечение поставок через цепь поставщиков

Этот вид деятельности относится к закупке такой продукции, как сырье, используемое в процессе производства на предприятии, или же товаров, распространяемых данным предприятием. Большинство компаний на сегодняшний день прибегают для этой работы к помощи телефона, факса или почты.

Например, рассмотрим некоторое вымышленное предприятие — "Volcano Cofee" — национальную сеть магазинов по продаже кофе и чая. Это предприятие вынуждено заказывать большое количество кофе в зернах в компании "Cascade Coffee Roasters", и чая — в фирме "Margo Tea Company". Компания "Cascade Coffee Roasters" может не иметь достаточных запасов для немедленного выполнения данного заказа, и возможно, закупает необходимую часть запаса у компании "East Indonesia Coffee Company". Этот сценарий является примером обеспечения поставок через цепь поставщиков и может быть расширен за счет вовлечения любого количества торговых партнеров.

Сценарий 3: сбыт без посредников

Компании пользуются сбытом без посредников при помощи Web, для того чтобы получить выгоду от спонтанных закупок, производимых клиентами.

Например, компания "1-800-FLOWERS" рекламирует свое присутствие в Internet на других популярных сайтах. Стимулом для владельцев этих сайтов являются комиссионные начисления от каждой продажи, связанной с их сайтами. Каждому владельцу Web-сайта или электронному партнеру (e-Partner) присвоен уникальный идентификатор (ID). К тому же, каждый заказ имеет идентификационный номер партнера (ePartnerID), относящийся именно к данному заказу. Это позволяет компании "1-800- FLOWERS" рассчитывать размер комиссионных начислений своим торговым партнерам.

Внимание!

EPartnerID — переменное имя.

Сравнение систем электронной коммерции классов business-to-business и business-to-consumer

Электронная коммерция класса business-to-consumer отражает опыт торговли реального мира в виде "электронных пассажей" (electronic malls) или виртуальных витрин (virtual storefronts), где пользователи делают покупки в онлайновом режиме, используя кредитные карты.

В электронной коммерции класса business-to-business транзакции осуществляются посредством частных сетей. Выполнение этих транзакций предназначается лишь для определенного круга "внутренних" заказчиков и бизнес-партнеров, а платежи производятся в соответствии с заранее определенными правилами кредитования или условиями аккредитива (credit terms).

Перечислим основные характеристики этих типов электронной коммерции.

Категория	Характеристика
business-to-consumer	❒ Основывается на Internet
	❒ Неограниченный доступ
	❒ Верифицированные платежи по кредитным картам
business-to-business	❒ Основывается на extranet
	❒ Допускается участие лишь бизнес-партнеров
	❒ Платежи осуществляются в соответствии с заранее определенными правилами кредитования
	❒ Электронные каталоги
	❒ Безопасность на основе firewall, применение паролей, аутентификация, шифрование, проверка полномочий
	❒ Выполнение транзакций посредством EDI

Пример Web-сайта

Web-сайт электронной коммерции класса business-to-business должен быть не только прост в использовании и безопасен, но также обязан предоставлять торговым партнерам такие преимущества, как низкие цены и высокую степень безопасности данных. Далее в этой главе мы сперва рассмотрим основные функциональные возможности такого Web-сайта. Затем вы познакомитесь с примером сайта электронной коммерции.

Требования к электронной коммерции класса business-to-business

Успешно функционирующий Web-сайт электронной коммерции класса business-to-business должен быть "дружелюбным" по своему дизайну, предоставлять пользователям необходимую информацию с минимальным временем ожидания и обеспечивать безопасность от нежелательных вторжений, а деловая информация обязана предоставляться только электронным партнерам (ePartners), имеющим соответствующие полномочия. Таким образом, реализация электронной коммерции должна отвечать определенным важным требованиям.

На рис. 1.4 показаны четыре главных требования, необходимых для успешного решения вопросов электронной коммерции класса business-to-business.

Рис. 1.4. Главные требования для успешного решения вопросов электронной коммерции

Электронные каталоги

Большинству предприятий, использующих электронную коммерцию, требуется создавать электронные каталоги, обладающие усовершенствованными возможностями параметрического поиска, помогающие электронным партнерам проводить быстрый поиск и сравнение необходимой продукции.

Доступ с соблюдением безопасности

Web-сайты электронной коммерции обязаны обладать свойствами, обеспечивающими безопасность информации, управляющими доступом к сайту клиентов, электронных партнеров и операторов.

Решения, основанные на стандартах

Предприятия должны вкладывать средства в открытые стандартные платформы и протоколы. Это снижает риск и создает гарантию максимального уровня взаимодействия между различными компонентами всей системы.

Более простой и дешевый обмен документами

Предприятиям необходимо обеспечивать взаимообмен структурированными бизнес-документами между торговыми партнерами на основе таких транзакционных наборов, как ANSI x.12.

Пример посещения Web-сайта

Целью данной демонстрации является ознакомление с примером Web-сайта и рассмотрение его следующих функциональных возможностей:

- вход на сайт в качестве покупателя (сайт FiveLakes);
- вход на сайт в качестве администратора или торгового партнера (сайт Ramona Publishing);
- закупка товара в онлайновом режиме (сайт FiveLakes);
- поиск на сайте с использованием различных параметров (сайт Ramona).

Примечание

Примеры сайтов не инсталлируются на компьютеры обучающихся.

Примечание

Вы можете создать Web-сайт, подобный данному примеру сайта, снабдив его перечисленными выше функциональными возможностями. Для того чтобы скопировать интерфейс примера сайта, обратитесь к *приложению А*. Можно модифицировать HTML-код ASP-страницы, автоматически сгенерированной мастером, для придания ему необходимого вида.

Для доступа на Web-сайт FiveLakes:

1. Запустите Microsoft Internet Explorer.
2. Введите адрес URL: **http://Instructor/FiveLakes**.

Для просмотра каталога продукции:

1. На домашней странице сайта FiveLakes под областью **Book Section** щелкните по изображению **Computers**.
2. Просмотрите книги, имеющиеся для данной категории.
3. На панели навигации нажмите кнопку **Home** для просмотра книг другой категории.

Для покупки на Web-сайте FiveLakes:

1. На домашней странице сайта FiveLakes под областью **Book Section** щелкните по изображению **Computers**.
2. На странице **Book Sections** в списке имеющихся книг щелкните по категории `Mastering Windows NT Server 4`.

   Обратите внимание на сопутствующую продукцию на странице **Product Information**.
3. На странице **Product Information** для выбранного элемента `Mastering Windows NT Server 4` щелкните на изображении **BUY NOW** (Купи сейчас), чтобы внести этот продукт в свою покупательскую корзину.
4. В поле **Qty** введите число `50`, а затем нажмите кнопку **Update Quantities** (Обновить количество).
5. На странице **Your Shopping Cart** щелкните по кнопке **Continue Checkout** (Продолжить выбор).
6. На странице **Shipping** (Отгрузка) введите информацию об отгрузке, а затем нажмите кнопку **Total** (Итого).
7. На странице **Payment** (Оплата) введите информацию о кредитной карте и номере счета. Вы можете воспользоваться следующими тестовыми номерами кредитных карт:
   - American Express: `3111-111111-11117`
   - Visa: `4111-1111-1111-1111`
   - MasterCard: `5111-1111-1111-1118`
   - Discover: `6111-1111-1111-1116`
8. Нажмите кнопку **Purchase** (Покупка) для подтверждения заказа. Обратите внимание, что на странице **Purchase Confirmation** (Подтверждение покупки) появится номер заказа.

Для того чтобы стать электронным партнером Web-сайта Ramona Publishing:

1. Запустите Microsoft Internet Explorer.
2. Введите адрес URL: **http://Instructor/Ramona**.
3. На домашней странице сайта щелкните ссылку **become an e-partner** (стать электронным партнером) для ввода информации о e-Partner.
4. На странице **e-Partner Registration Form** (Регистрация электронного партнера) введите необходимую информацию и затем щелкните кнопку **Save** (Сохранить).

Для получения доступа на Web-сайт Ramona Publishing в качестве администратора в Microsoft Internet Explorer введите адрес URL: **http://Instructor /Ramona/Manager**.

Для добавления электронных партнеров на Web-сайте:

1. На странице **Site Manager: Ramona Publishing** нажмите кнопку **e-Partners**.
2. На странице **e-Partner** щелкните мышью по имени организации электронного партнера в таблице **Prospective e-Partners** (Предполагаемые электронные партнеры) для просмотра информации об электронном партнере.
3. На странице **Prospective e-Partner Information** (Информация о предполагаемых электронных партнерах) щелкните по ссылке **Add to e-Partner List** (Добавить в список электронных партнеров), чтобы включить выбранного в имеющемся списке партнера в список электронных партнеров Web-сайта.

   Обратите внимание, что название организации электронного партнера появилось в таблице **Existing e-Partners** (Существующие электронные партнеры).

Для получения доступа на Web-сайт Ramona Publishing в качестве электронного партнера:

1. В Microsoft Internet Explorer введите адрес URL: **http://Instructor/Ramona**.
2. На домашней странице Web-сайта введите в соответствующих текстовых полях регистрационный идентификатор (login ID) и пароль электронного партнера, а затем нажмите кнопку **Login** (Регистрация).

Для поиска продукции на Web-сайте Ramona Publishing:

1. На домашней странице Web-сайта нажмите кнопку **Search**[1].
2. В области **Search** в текстовом поле введите `Windows NT`, а затем нажмите кнопку **Find**.
3. Просмотрите результаты поиска.

Для просмотра заказа на покупку на Web-сайте Ramona Publishing:

1. На домашней странице введите URL-адрес: **http://instructor/Ramona /Manager** для перехода на страницу Manager.
2. Нажмите кнопку **Purchase Order** (Заказ на покупку).

   На этой странице отображается заказ на покупку, инициированный Web-сайтом FiveLakes и полученный сайтом Ramona Publishing.

---

[1] Кнопка **Search** находится на навигационной панели Internet Explorer. — *Ред*

Безопасность на Web-сайте электронной коммерции

Безопасность является важным аспектом любого Web-сайта электронной коммерции класса business-to-business. Ни одно предприятие не заинтересовано в разглашении информации, предназначенной для строго определенного круга сотрудников, следовательно, необходимо, чтобы такая информация была доступна только тем, кто имеет на это соответствующие полномочия. Например, покупателю не понравится, если номер его кредитной карты или другая персональная информация станут известны третьей стороне, которая может воспользоваться ею в корыстных целях.

В *разд. "Обзор протоколов SSL и SET"* вы узнаете о протоколах, которые используются для обеспечения безопасности транзакций.

В *разд. "Методы защиты"* вы познакомитесь с методами обеспечения безопасности информации на Web-сайте.

Обзор протоколов SSL и SET

На Web-сайте электронной коммерции для обеспечения безопасности транзакций business-to-business (компания-компания) и business-to-consumer (компания-заказчик) используются различные типы протоколов. Эти протоколы служат для осуществления разных типов безопасных транзакций. Ниже приводятся наиболее распространенные протоколы.

Протокол Secure Sockets Layer

SSL (Secure Sockets Layer, протокол безопасных соединений) является протоколом безопасности, служащим для передачи частных документов в Internet. В SSL применяется личный ключ для шифрования данных, передаваемых по SSL-соединению. Этот протокол обеспечивает шифрование данных, аутентификацию сервера, целостность сообщений, а также имеет факультативную возможность аутентификации клиента для соединения TCP/IP.

HTTP (Hypertext Transfer Protocol, протокол передачи гипертекста) предназначен для передачи данных в Internet. HTTP может использовать SSL для создания безопасных соединений между клиентом и сервером, по которым могут быть переданы любые объемы защищенных данных. Такая комбинация HTTP и SSL называется HTTPS или HTTP Secure. Адреса Web-страниц, использующих HTTPS, начинаются с https:// вместо http://.

Протокол Secure Electronic Transaction

SET (Secure Electronic Transaction, защищенные электронные транзакции) — это стандарт, который позволяет защищать сделки, совершаемые в Internet

с помощью кредитных карт. Используя цифровые подписи, SET позволяет торговцам осуществлять проверку (верифицировать) возможных покупателей. Он также защищает покупателей с помощью механизма, благодаря которому номера их кредитных карт способны передаваться непосредственно производителю электронных карт для их верификации и перечисления сумм, подлежащих выплате (billing). Таким образом, торговец никогда не видит номер кредитной карты покупателя.

Внимание!

SET — это спецификация, которая продолжает развиваться. Так как некоторые реализации SET не соответствуют данной спецификации, SET не получил широкого распространения.

Методы защиты

В электронной коммерции класса business-to-business обычно принимают участие известные друг другу торговые партнеры. Однако при передаче уязвимых данных в сети любого типа очень важно не только аутентифицировать отправителя и получателя, но и обеспечить целостность и секретность самих данных. Меры по обеспечению безопасности включают в себя аутентификацию бизнес-транзакций, управление доступом к таким ресурсам, как Web-страницы, зарегистрированных или выбранных пользователей, шифрование сообщений, и проверку секретности и эффективности транзакций.

На рис. 1.5 представлены четыре элемента эффективной безопасности.

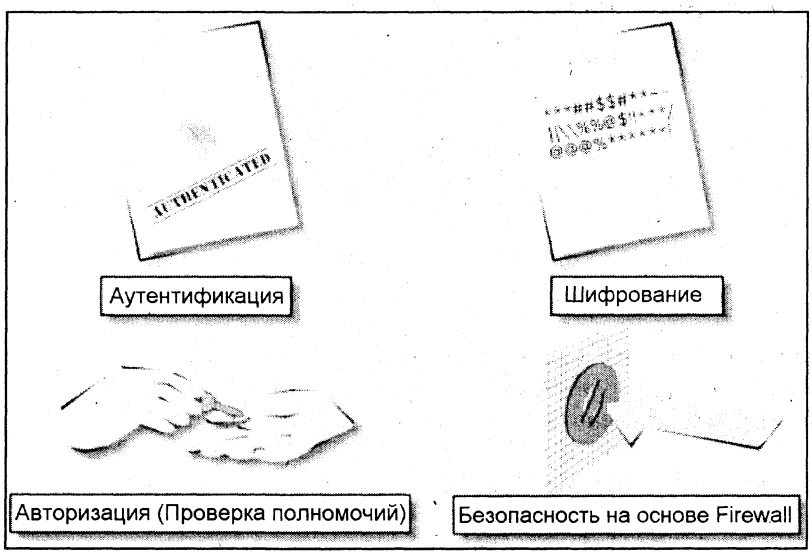


Рис. 1.5. Элементы эффективной безопасности

Аутентификация

Одним из наиболее важных факторов, влияющих на безопасность электронной коммерции, является аутентификация. *Аутентификация* предотвращает неавторизованный доступ к информации ограниченного распространения и обычно осуществляется при помощи паролей. При подключении пользователя у него запрашивается регистрационный идентификатор (имя для регистрации) (login ID) и пароль, которые позволяют ему получить доступ к частному Web-сайту продавца.

Другой уровень безопасности предоставляется пользователям, которые хотят скачать по сети (download) цифровой сертификат с Web-сайта продавца на компьютер, который они будут использовать для подключения к данному сайту. Применение цифрового сертификата гарантирует, что данное соединение шифруется с помощью протокола SSL. К тому же, сервер продавца верифицирует компьютер пользователя по мере того, как пользователь получает доступ к учетной записи (бюджету) сайта (site's account).

Шифрование

Шифрование обеспечивает, что информация, передаваемая по сети, может быть прочитана и модифицирована только авторизованными пользователями. В данном методе по протоколу SSL шифруется информация, передаваемая между компьютером пользователя и сервером продавца, тем самым обеспечивается целостность передаваемых данных.

Авторизация

Системы *авторизации* гарантируют, что неавторизованные пользователи не смогут получить доступ к файлам и данным, предназначенным для ограниченного распространения. Эти системы обычно требуют однократной регистрации пользователя, а затем предоставляют доступ ко всем приложениям и компьютерам, на работу с которыми у пользователя есть соответствующие полномочия.

Безопасность на основе Firewall

Firewall (брандмауэр) — это совокупность программного и аппаратного обеспечения для контроля, фильтрации и защиты внутренней корпоративной сети или intranet от неавторизованного доступа. Брандмауэры часто представляют собой аппаратные системы фильтрования, установленные на каждом входе или стыке передачи во внутреннюю сеть. Специалисты по безопасности в Internet и intranet полагают, что использование брандмауэров может остановить до 90% попыток неавторизованного доступа.

Создание группы электронных партнеров

Одним из методов обеспечения безопасности является аутентификация пользователей Web-сайта. При помощи аутентификации, лишь авторизованные пользователи — обычно, электронные партнеры (e-Partners) — могут получать доступ к важной деловой информации, такой как, например, сведения о ценах и наличии изделий.

Далее вы узнаете о концепции группы электронных партнеров, а также создадите группу электронных партнеров для Web-сайта.

Что такое группа электронных партнеров?

Предприятие, занимающееся электронной коммерцией, не имеет электронного взаимодействия с каждой из групп, желающих заниматься с ним совместным бизнесом. Вместо этого оно осуществляет транзакции лишь с доверенными деловыми сторонами, по своему выбору, именуемыми электронными партнерами (e-Partners).

Для того чтобы стать электронным партнером, предприятие обязано сначала обратиться с соответствующей просьбой (заявлением) к тому предприятию (организации), с которым оно желает сотрудничать, и затем это предприятие (организация) рассматривает просьбу заявителя. Если заявитель удовлетворяет необходимым требованиям, то он включается в списки электронных партнеров.

Электронным партнерам предоставляется привилегированный доступ к некоторым страницам, которые скрыты от неавторизованных пользователей Internet.

Создание группы электронных партнеров на практике

В данном практическом разделе мы разрешим посетителю сайта стать электронным партнером Web-сайта Ramona Publishing. Вы также проведете аутентификацию электронных партнеров, проверив действительность их регистрационных идентификаторов (имен) и паролей прежде, чем они смогут получить доступ к сайту.

В представленной ниже процедуре вам нужно модифицировать страницу по умолчанию — default.asp, которая отображает страницу регистрации (login page), предназначенную для ввода регистрационного идентификатора и пароля электронного партнера. Эта страница также должна обеспечивать ссылку на страницу регистрации нового партнера.

Для отображения страницы с регистрационным именем:

1. Скопируйте содержимое папки Practices\Mode01\Start — вместе с подкаталогом Manager — с CD-ROM в папку C:\InetPub\wwwroot\Ramona. В папке Start находятся файлы страниц default.asp, home.asp, newuser.asp и new_reg.asp. Если появится предупреждение о перезаписи, нажмите кнопку **Yes**.
2. В меню кнопки **Start** выберите **Programs |Windows NT 4.0 Option Pack | Microsoft Script Debugger**, а затем укажите **Microsoft Script Debugger**.
3. Откройте страницу default.asp в папке C:\InetPub\wwwroot\Ramona.
4. Модифицируйте код для того, чтобы обеспечить отправку содержания на форму `frmsignon`, добавив атрибут `ACTION` следующим образом:

```
<form name="frmsignon" METHOD="POST" ACTION="validate.asp">
```

5. Страница validate.asp будет содержать код для проверки регистрационного идентификатора (login ID) и пароля электронного партнера.
6. Перед тегом `</form>` вставьте код для обеспечения ссылки на страницу newuser.asp следующим образом:

```
<table>
  <tr>
    <td><a href="newuser.asp"> Become an e-Partner </a></td>
  </tr>
</table>
```

7. Страница newuser.asp содержит код, служащий для приема информации о пользователе, который пожелал стать электронным партнером, а также для отправки этой информации в файл register.asp.
8. Сохраните страницу default.asp.

Для того чтобы сохранить информацию о предполагаемом электронном партнере в таблице базы данных:

1. В Microsoft Script Debugger откройте файл register.asp.
2. Следующий код служит для открытия таблицы `ramona_epartner` в базе данных `Ramona`:

```
<%
  set con=server.createobject("adodb.connection")
  set rs=server.createobject("adodb.recordset")
  con.open "database=ramona;dsn=ramona;uid=sa;password=;"
  rs.open "ramona_epartner",con,2,2
```

3. Представленный ниже код позволит запомнить информацию о пользователе в таблице `ramona_epartner`.

```
rs.addnew
rs("login_id")=request("lid")
rs("password")=request("pwd")
rs("lastname")=request("lname")
rs("firstname")=request("fname")
rs("company_name")=request("cname")
rs("billing_contact")=request("billc")
rs("address")=request("address")
rs("city")=request("city")
rs("state")=request("state")
rs("zip")=request("zip")
rs("country")=request("country")
rs("phone")=request("phone")
rs("email")=request("email")
rs("partner")=0
rs.update
rs.close
con.close
```

Примечание

В данном фрагменте кода `request` относится к `request.form`.

4. Следующий код служит для отображения пользователю страницы подтверждения:

```
   response.redirect "new_reg.asp"
%>
```

   Страница new_reg.asp создана.

5. Закройте страницу register.asp.

Для проверки регистрационного идентификатора и пароля электронного партнера:

1. В Microsoft Script Debugger откройте файл validate.asp.
2. Следующий фрагмент кода служит для открытия базы данных `Ramona`:

```
<%
  set con=server.createobject("adodb.connection")
  set rs=server.createobject("adodb.recordset")
  con.open "database=ramona;dsn=ramona;uid=sa;password=;"
```

3. Представленный ниже фрагмент кода обеспечит поиск значений регистрационного идентификатора и пароля на форме `frmsignon` страницы default.asp:

```
  user=request.form("login_id")
  pwd=request.form("password")
```

4. Данный фрагмент кода служит для того, чтобы проверить наличие значений регистрационного идентификатора и пароля в таблице ramona_epartner:

```
sqlstr= "select login_id, password, partner from "& _
"ramona_epartner where login_id='" & user & "'" & _
"and (password='" & pwd & "'" & " and partner=1)
set rs=con.execute(sqlstr)
```

5. Следующий фрагмент кода позволит электронному партнеру перейти на домашнюю страницу в том случае, если получено соответствие. Иначе, будет произведен переход на страницу default.asp.

```
  if rs.eof then
    response.redirect "default.asp"
  else
    response.redirect "home.asp"
  end if
%>
```

6. Закройте страницу validate.asp.

Для того чтобы стать электронным партнером:

1. Запустите Microsoft Internet Explorer.
2. Введите адрес URL: **http://localhost/Ramona**.
3. На странице регистрации (login page) щелкните ссылку **Become an e-Partner** для ввода информации об электронном партнере.
4. На странице регистрации электронных партнеров **e-Partner Registration** введите информацию об электронном партнере и затем нажмите кнопку **Submit** (Отправить).

Для того чтобы добавить электронного партнера на Web-сайте:

1. В Microsoft Internet Explorer введите адрес URL: **http://localhost/Ramona /Manager**.
2. На странице **Site Manager: Ramona Publishing** нажмите кнопку **e-Partners**.
3. На странице **e-Partner** выберите имя электронного партнера в таблице **Prospective e-Partners** (Предполагаемые электронные партнеры) для просмотра информации об электронном партнере.
4. На странице **Prospective e-Partner Information** (Информация о предполагаемых электронных партнерах) щелкните по ссылке **Add to e-Partner List** (Добавить в список электронных партнеров) для включения электронного партнера в существующий список электронных партнеров на Web-сайте.

Для получения доступа к Web-сайту в качестве электронного партнера:

1. В Microsoft Internet Explorer введите адрес URL: **http://localhost/Ramona**.
2. На странице регистрации укажите регистрационный идентификатор и пароль электронного партнера в соответствующих текстовых полях и затем нажмите кнопку **Sign On**.

Вопросы для повторения

1. Укажите достоинства электронной коммерции класса business-to-business[1].
2. Назовите методы, которые могут использоваться для установления безопасных транзакций.
3. Назовите требования, предъявляемые к Web-сайту для эффективной электронной коммерции класса business-to-business.

[1] Ответы на вопросы к каждой главе см. в конце книги. — *Ред.*

ГЛАВА 2

Формирование каталогов и поиск

Краткий обзор

В электронной коммерции важно, чтобы посетители Web-сайта компании имели возможность просмотра в онлайновом режиме каталога изделий или услуг, предлагаемых фирмой. Каталог должен включать в себя средства поиска, наподобие программы Microsoft Search, позволяющей посетителям быстро находить информацию о конкретном изделии или услуге. Вы можете настраивать средство поиска так, чтобы ускорить процесс нахождения необходимой информации в каталоге и при этом получать более точные результаты.

Цели и задачи

После прочтения данной главы вы научитесь:

- пояснять функциональные возможности программы Microsoft Search;
- описывать, как работает данная программа;
- создавать каталог с помощью сервера формирования каталога (Catalog Build Server);
- осуществлять поиск в каталоге с помощью сервера Search;
- настраивать страницу поиска, включив новые поисковые возможности;
- создавать для пользователей ссылку на поиск каталога.

Примечание

Код, представленный в данной главе, имеется на CD-ROM в файле DemoCode\Mod02\Mod02Code.txt.

Программа Microsoft Search

Типичное предприятие (учреждение) работает с информацией о собственных изделиях, услугах и операциях, содержащейся во множестве различных документов. Если вы не знаете точного местоположения необходимых документов, вам будет весьма сложно отыскать нужную информацию.

Однако вы можете упростить эту работу, воспользовавшись программой Microsoft Search, которая позволяет сформировать каталог с индексированными документами и удобным средством поиска для посетителей сайта.

В *разд. "Функции Microsoft Search"* вы ознакомитесь с применением, возможностями и требованиями данной программы.

В *разд. "Как работает Microsoft Search"*, вы узнаете о процессе создания каталога со встроенным в него средством поиска.

Функции Microsoft Search

Предположим, вы являетесь изготовителем и желаете, чтобы все ваши изделия были представлены в онлайновом каталоге. Ваш каталог должен включить утилиту типа Microsoft Search, позволяющую вашим клиентам и электронным партнерам осуществлять быстрый поиск необходимой информации. Вы можете использовать Microsoft Search для:

- создания каталога изделий, услуг или информации;
- копирования каталога на различные поисковые компьютеры (хосты);
- обеспечения поисковых возможностей вашего каталога.

Преимущества Microsoft Search

В электронной коммерции класса business-to-business легкость доступа к информации для вас и ваших электронных партнеров является обязательным требованием. Размещение этой информации может быть затруднено, если она храниться в различных базах данных, документах, файловых системах и на разных серверах.

Вы можете существенно упростить процесс поиска, создав индексированный каталог информации, используя для этого программу типа Microsoft Search.

Требования к ресурсам

Для формирования каталогов и осуществления поиска в них в режиме on-line необходимо использовать Windows NT Server или один из программных продуктов семейства Windows 2000, а также установить Microsoft Search.

Сервер действует как главный поисковый компьютер (поисковый хост) (search host), позволяющий:

- формировать каталог при помощи Catalog Build server;
- осуществлять поиск в каталоге, используя для этого сервер Search.

Хотя система Search может содержать много хостов, по крайней мере, один из них должен быть сконфигурирован для формирования онлайного каталога. Тот же самый или же другой хост может быть сконфигурирован для хранения каталога и поиска в нем. Каждый хост может быть настроен с учетом системных ресурсов, размещения файлов и т. д.

Как работает Microsoft Search

В процессе формирования каталога Microsoft Search осуществляет:

- подбор содержимого;
- выделение информации;
- создание индекса каталога;
- компилирование и распространение каталога;
- предоставление пользователям возможности обращения к странице поиска.

Типы каталогов

Имеются три различных типа каталогов: Crawl (с использованием поисковых серверов), Notification (с использованием уведомляющих серверов), и Database (с использованием доступа к базе данных). Тип каталога определяется тем методом, который в нем применяется для сбора информации (табл. 2.1).

Таблица 2.1. *Типы каталогов*

Тип каталога	Описание
Crawl-каталог	Построен при помощи поиска документов в Internet, intranet, файловых системах, в папках Microsoft Exchange с общим доступом (Public)
Notification-каталог	Построен на основе информации, полученной от уведомляющих серверов
Database-каталог	Построен в результате просмотра таблицы базы данных с помощью ODBC (Open Database Connectivity, интерфейс открытого взаимодействия с базами данных)

Подбор содержимого

Подбор (Gathering) — процесс сбора документов, предназначенных для включения в каталог. Microsoft Search использует адрес Web-страницы или

путь к каталогу в качестве начального адреса для подбора документов. Наиболее распространенным методом подбора является crawling-метод. В нем можно выделить три различных типа (табл. 2.2).

Таблица 2.2. *Типы crawling-метода подбора содержания*

Типы crawling-метода подбора содержимого	Описание
Последовательный обход ссылок в Web (Web link crawl)	Применяется для сбора информации в Web. Microsoft Search осуществляет сбор ссылок, содержащихся на начальной странице поиска, и последовательно обходит те страницы, на которые указывают эти ссылки
Последовательный обход файлов (File crawl)	Используется для подбора информации из документов, размещенных в каком-либо каталоге. Microsoft Search осуществляет последовательный обход каталога, указанного в начальном адресе, и собирает все документы, хранящиеся в каталоге и его подкаталогах. Такой последовательный просмотр файлов позволяет придерживаться списка управления доступом к файлам в NFTS
Последовательный обход сообщений Microsoft Exchange	Используется для поиска на основе сообщений сервера Microsoft Exchange. Начальным адресом всегда является папка Public в Exchange. Права на документы в данной папке сохраняются

Отслеживание документов для отбора

Когда Microsoft Search начинает *последовательный обход* (crawling), он создает внутренний файл (журнал) регистрации, для отслеживания всех отобранных документов. Этот файл регистрации называется *журналом транзакции* (transaction log). Microsoft Search хранит ссылки на все файлы, сообщения, а также Web-ссылки, полученные по начальному адресу, и отмечает их как "отложенные" (pending). После того как отбор документа будет успешно завершен, Microsoft Search отмечает ссылку на него в журнале транзакции как "обработанную" (done).

Сохранение истории последовательного обхода

Осуществляя последовательный обход документов, Microsoft Search ведет запись всех имен файлов, которые были пройдены. Эта запись (отчет) называется *историей последовательного обхода* (crawl history). Каждая встретившаяся новая ссылка вносится в журнал транзакции, но лишь после того, как

будет осуществлено сравнение данной ссылки с историей последовательного обхода. Это гарантирует, что никакая ссылка не будет пройдена повторно.

Указание границ просмотра

Вы можете ограничивать число ссылок, вносимых в журнал транзакции, путем точного указания в программе Microsoft Search требуемой глубины обхода (поиска). Это можно сделать следующими способами:

- можно ограничить уровень подкаталогов при обходе;
- можно установить допустимое число переходов со страницы на страницу или с сайта на сайт, производимое в процессе последовательного обхода;
- при страничном переходе, Microsoft Search может осуществлять переход в любом направлении в пределах сайта, начиная с начального адреса;
- при переходе с сайта на сайт Microsoft Search пересекает границы сайта в процессе последовательного обхода, начиная с начального адреса;
- можно устанавливать правила, действующие при выполнении последовательного обхода Web-ссылок. С помощью этих правил вы способны указать путь, которому Microsoft Search будет следовать.

Сравнение полного и инкрементного обхода

При первом последовательном обходе Microsoft Search выполняет так называемый *полный обход* (full crawl). Однако этот метод занимает много времени и требует большой пропускной способности, так что он не является целесообразным для обновления существующего каталога. Вместо этого, вы можете использовать так называемый *инкрементный* (*избирательный* или *ступенчатый*) *обход*, при котором Microsoft Search начинает с предыдущего каталога и далее обходит лишь те документы, которые были изменены. Это существенно снижает время и пропускную способность, необходимую для обновления каталога.

Извлечение информации

Выполняя подбор документов, Microsoft Search открывает каждый документ и использует фильтры для сбора информации в виде текстового содержания (content), ссылок и определенных свойств документов. Эта программа поддерживает фильтры для распространенных форматов файлов, таких как HTML, документов Microsoft Office, файлов простого текстового формата, а также фильтры третьих фирм для извлечения информации из документов других типов.

Microsoft Search может автоматически определять языки, используемые в собранных документах. Эта особенность помогает в применении соответствующих отличительных признаков слов (word-breakers), служащих для иден-

тификации отдельных слов и соответствующих лингвистических основ, необходимых для производства грамматически правильных вариантов слов.

Создание индекса каталога

Microsoft Search использует информацию, извлеченную из собранных документов, чтобы создать индекс слов, присутствующих в данных документах, с указанием их местоположения. Несущественные слова, например, артикли, опущены.

Формирование и распространение каталога

После извлечения и индексации информации из собранных документов Microsoft Search собирает ее в каталог. Когда каталог формируется, Microsoft Search копирует его на хосты, включенные в список хостов поискового сервера. Этот процесс называется *распространением* (propagating).

Предоставление посетителям сайта возможностей поиска в каталоге

После того как каталог создан, заданная по умолчанию страница поиска расположена по адресу URL: **http://Server_name/Siteserver/Knowledge/Search/**. Посетителям сайта необходимо предоставить доступ к этой странице, чтобы разрешить поиск в каталоге.

Примечание

Для каталогов, которые построены из базы данных, заданная по умолчанию страница поиска расположена по URL **http://localhost/siteserver/knowledge /search/database/Search /catalog_name/search.htm**.

Создание каталога при помощи Catalog Build Server

Чтобы сформировать каталог, прежде необходимо подготовить информацию и набор инструкций, соответствующих типу создаваемого вами каталога. После формирования каталога, вы можете его скопировать или распространить на различные поисковые хосты. Вы можете также задать режим автоматической перестройки каталога по расписанию. В следующем разделе приводится подробное пояснение этих возможностей.

Описание каталога

Перед формированием каталога любого типа необходимо сформулировать (построить) определение каталога, основываясь на том, к какому типу он относится.

Элементы определения каталога

Определение каталога должно содержать информацию, подробно описанную в табл. 2.3.

Таблица 2.3. *Информация, содержащаяся в определении каталога*

Необходимая информация	Описание
Имя каталога	Каждый каталог должен иметь уникальное имя
Начальный адрес и стратегия обхода	Начальным называется адрес, с которого Microsoft Search производит обход. В качестве стартового адреса может быть адрес Web-страницы, некоторой папки или папки Public в Microsoft Exchange. Стратегия обхода (crawling policy) указывает направление обхода от начального адреса
Правило, определяющее сайт и путь	Правило, определяющее сайт, указывает, какие сайты необходимо обойти, а какие следует пропустить. Правила могут быть настроены так, чтобы определенные пути на Web-сайте можно было пропустить при обходе
Типы файлов	Указывают, файлы каких типов должны быть собраны для каталога
Список распространения	Указывает, на какие сайты должен быть отправлен данный каталог
Установки (параметры) журнала ошибок сбора информации (gatherer log)	Каждый раз при формировании каталога создается журнал ошибок, в который заносятся любые ошибки, возникшие при обращении Microsoft Search к документам
Расписание формирования каталога	Служит для автоматического формирования каталога
Распределение доступа и отображения	Применяется в тех случаях, когда информация должна быть собрана по одному адресу, а отображена по другому адресу. Эта возможность позволяет использовать преимущества системы целостности файлов в Microsoft Windows NT при поиске файлов

Можно создать определение каталога при помощи Microsoft Management Console (MMC) или Web Based Administration (WebAdmin) на Microsoft Commerce Site Server.

Распространение каталога

Распространение — процесс копирования каталога на поисковые серверы его ведущих (главных) сайтов, после того, как он будет сформирован. Ведущими называются такие сайты (host sites), где посетители могут просматривать каталог и осуществлять в нем поиск. Ведущие сайты указаны в определении каталога. Любой каталог может быть скопирован (распространен) максимум на 32-х хостах, включая тот, на котором он был первоначально сформирован.

Поисковый сервер может получить каталог только в том случае, если на сервере установлена и включена в систему программа Microsoft Search. Обратите внимание, что административная учетная запись доступа должна содержать административные права и на Catalog Build Server, и на тот хост, куда будет скопирован данный каталог.

В то время как каталог формируется, он может быть разослан на хосты после того, как в него будет включено некоторое количество документов. В таком случае, на хостах будет размещен только частично заполненный каталог. Каталог регулярно обновляется до тех пор, пока не будет полностью сформирован. Затем он рассылается на хосты.

Создание определения каталога из базы данных

Каталог базы данных создается на основе обхода базы данных с ODBC-доступом. Например, вы можете хранить статьи и доклады в таблице базы данных, каждая запись которой содержит имя автора, дату публикации, описание, и непосредственно содержание. Вы можете создать такой каталог, который предоставит посетителям сайта доступ к этой информации и возможность осуществлять в ней поиск.

На рис. 2.1 представлены три столбца, необходимые для определения каталога.

Определения каталога на основе базы данных могут быть созданы с помощью Create New Catalog Definition Wizard (Мастера создания определений новых каталогов), который входит только в состав Web Based Administration (WebAdmin).

Чтобы создавать определение для каталога базы данных, необходимо:

- указать источник данных ODBC, выбирая его из списка системных источников данных;
- указать уникальное имя каталога;
- указать таблицу, используемую в каталоге;

- определить столбец базы данных, используемый для содержания. Этот столбец отображается, когда посетитель сайта щелкает по ссылке получения результата;
- определить столбец базы данных, используемый для первичного ключа. Этот столбец содержит уникальные значения для каждой строки в данной таблице;
- определить столбец базы данных, используемый для заголовка. Этот столбец отображается в качестве заголовка, когда посетитель сайта щелкает по ссылке получения результата.

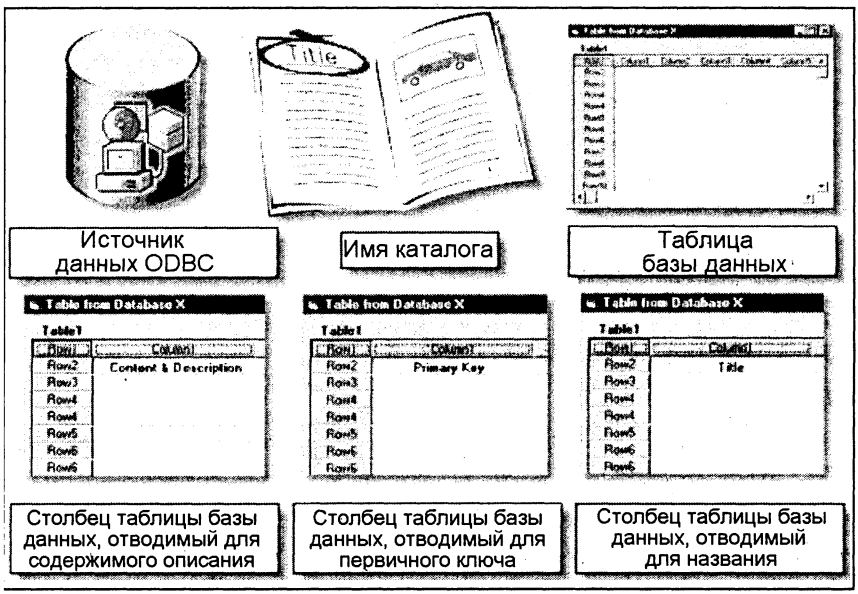


Рис. 2.1. Столбцы, необходимые для определения каталога

Используя информацию, представленную в определении каталога, мастер Create New Catalog Definition Wizard (Мастер создания определений новых каталогов) формирует все необходимые активные серверные страницы (asp) и затем строит каталог. Создаются страницы, перечисленные в табл. 2.4.

Таблица 2.4. *Перечень ASP*

Страница	Описание
Dir.asp	Используется для хранения начального (стартового) адреса каталога
Row.asp	Применяется для поиска каждой записи в базе данных

Таблица 2.4 (окончание)

Страница	Описание
View.asp	Служит для просмотра каждой записи
Search.htm	Предназначен для сбора ключевых слов поиска
Results.asp	Применяется для отображения результатов. Каждая ссылка, представленная на этой странице, использует View.asp

Обращение к поисковой странице осуществляется по адресу URL: **http://localhost/siteserver/knowledge/search/database/search/catalog_name/search.htm**. Пользователям необходимо предоставить ссылку на эту страницу, чтобы они могли осуществлять поиск в данном каталоге.

Пример формирования каталога

В этом демонстрационном примере вы создадите онлайновый каталог при помощи Catalog Build Server из состава Web Based Administration (WebAdmin). Microsoft Search будет обходить базу данных, размещенную на поисковом хосте для того, чтобы осуществить поиск информации, хранящейся в базе данных.

Для запуска Web Based Administration (WebAdmin):

1. В меню кнопки **Start** выберите команду **Programs**.
2. Укажите пункт **Microsoft Site Server**, а затем команду **Administration**.
3. Выберите **Site Server Service Admin (HTML)**.

Для создания каталога базы данных при помощи WebAdmin:

1. В окне **Web Based Administration** нажмите кнопку **Search**.
2. В левой части щелкните **Catalog Build Definition** (Определение для построения каталога).
3. В окне **Catalog Build Definition** (Определение для построения каталога) нажмите кнопку **Create**.
4. В окне **Create New Catalog Definitions Wizard** (Мастер создания определения нового каталога) укажите источник данных для каталога. В качестве источника данных вы будете использовать базу данных. Выберите **A database** (База данных) и нажмите кнопку **Next**.
5. В окне **Select Data Source and Specify Catalog Name** (Выбор источника данных и задание имени каталога) укажите базу данных и имя каталога. В поле **ODBC Data Source Name (DSN)** (Имя источника данных ODBC) выберите **FiveLakes**.

6. В поле **Catalog name** (Имя каталога) введите `FiveLakes`.
7. В поле **User name** (Имя пользователя) введите `sa`.
8. Поле для пароля (password) оставьте пустым и нажмите кнопку **Next**.
9. В окне **Select Table** (Выбор таблицы) выберите **FiveLakes_product** из раскрывающегося списка **Table** (Таблица), а затем щелкните по кнопке **Next**.
10. В окне **Select Content, Primary Key, and Hyperlink Columns** (Выбор столбцов содержания, первичного ключа и гиперссылок) в области **Content column** (Столбец содержимого) выберите столбец **description** (описание) базы данных.
11. В столбце **Primary key** (Первичный ключ) столбец **sku** указан по умолчанию. В области **Hyperlink column** (Столбец гиперссылок) выберите столбец **name** (имя), а затем нажмите кнопку **Next**.
12. Окно **Select Searchable and Retrievable Columns** (Выбор столбцов, по которым будет осуществляться поиск и выборка) позволяет вам определить те столбцы, которые в каталоге станут использоваться в качестве индексных при выборке и поиске. Воспользуйтесь табл. 2.5 для того, чтобы указать столбцы, участвующие в поиске и выборке в базе данных. Установите соответствующие флажки и щелкните по кнопке **Next**.

Таблица 2.5. *Использование столбцов*

Имя столбца	Используется при поиске	Используется при выборке
`sku`	Да	—
`list_price`	—	—
`image_file`	—	—
`image_width`	—	—
`image_height`	—	—
`sale_price`	Да	Да
`sale_start`	Да	Да
`sale_end`	Да	Да
`author`	Да	Да
`edition`	Да	Да
`publisher`	Да	Да
`category`	Да	Да

13. В окне **Enable Immediate Build** (Разрешить формирование) щелкните **Build the database catalog now** (Сформировать каталог базы данных).

14. Нажмите кнопку **Finish** для начала формирования каталога. В окне **Summary** (Итоги) отобразятся созданные файлы. Скрипты индексирования базы данных, dir.asp и row.asp созданы в папке <каталог_установки>\siteserver\knowledge\search\database\in-dex\FiveLakes\. Microsoft Search также создает скрипты поиска в базе данных — search.htm, results.asp и view.asp — в папке <папка_установки>\siteserver\knowledge\search\database\search\FiveLakes\. Доступ к странице поиска осуществляется по URL: **http://localhost/site-server/knowledge/search/database/search/FiveLakes/search.htm**.

15. Нажмите кнопку **OK**.

Пример создания расписания автоматического формирования каталога

Для каталога может быть задано расписание, по которому станет осуществляться его автоматическое формирование. Вы можете указать, будет ли это формирование полным или выборочным (ступенчатым).

В этом демонстрационном примере вы составите расписание для еженедельного автоматического обновления каталога сайта FiveLakes.

Для установки расписания автоматического формирования:

1. В окне **Catalog Build Definitions** выберите **FiveLakes**, а затем нажмите кнопку **Properties**.

2. В окне **Properties of FiveLakes** щелкните по кнопке **Build Schedule** (Составить расписание).

3. В окне **Build Schedule for FiveLakes** нажмите кнопку **Create**.

4. В окне **Select Task Action and Frequency** щелкните **Weekly** (Еженедельно) и затем нажмите кнопку **Next**.

5. В окне **Set Task Time and Date** выберите **Sunday** (Воскресенье), а затем нажмите кнопку **Next**.

6. В окне **Set Account Information** вам необходимо указать пароль и подтвердить его. Однако в данном демонстрационном примере, оставьте поля **Password** и **Confirm** пустыми. Щелкните по кнопке **Finish** для установки расписания автоматического формирования данного каталога.

Вы вернулись в окно **Build Schedule for FiveLakes**. Обратите внимание, что задача, которую вы указали, отображается на странице.

Поиск в каталоге с помощью Search Server

Каждый раз, когда вы создаете онлайновый каталог, вам нужно тестировать его, тщательно осуществляя в нем поиск различных данных. Вы можете настроить поисковые функции каталога так, чтобы предоставить посетителям сайта возможность более быстрого и точного получения результатов. Наконец, вам необходимо включить поисковую страницу в состав своего сайта.

Пример поиска в каталоге

В данном демонстрационном примере вы проведете поиск в каталоге FiveLakes тех записей, которые содержат некоторое заданное слово. Поисковая страница для этого каталога находится по URL: **http://localhost /siteserver/knowledge/search/database/search/FiveLakes/**.

Для поиска в каталоге базы данных:

1. Нажмите кнопку **Start**, а затем выберите команду **Run**.
2. В поле **Open** введите **http://*localhost*/siteserver/knowledge/sea-rch/database /search/FiveLakes/search.htm** и нажмите кнопку **OK**. Появится страница **Database Search: FiveLakes**. Вы можете использовать ее для поиска содержимого каталога сайта FiveLakes.
3. В поле **Enter text to search for** (Введите текст для поиска) введите `Microsoft`, а затем нажмите кнопку **Search**.
4. На странице **Search Results** (Результаты поиска) отображаются записи базы данных, содержащие слово Microsoft. Щелкните на элементе **Microsoft VC++ 6.0 Reference Library** для просмотра подробной информации этой книги.

Анализ программного кода поисковых страниц

Критически анализируя программный код, вы усовершенствуете поисковую страницу таким образом, чтобы предоставить возможность посетителям сайта отыскать книги, опубликованные двумя издателями, Microsoft Press и Prentice Hall.

Когда вы формируете каталог из базы данных, Microsoft Search создает страницы search.htm, results.asp и view.asp. В этом упражнении вам нужно будет модифицировать страницы search.htm и results.asp для того, чтобы обеспечить новые поисковые возможности для посетителей сайта.

На странице search.htm данные, вводимые пользователем, помещаются в текстовое поле **qu**. Для того чтобы посетители сайта могли осуществлять по-

иск книги определенного издателя, вы можете внести изменения в программный код страницы search.htm так, чтобы на ней отображался элемент combo box (поле со списком), содержащий определенные названия издательств, в данном случае — Microsoft Press и Prentice Hall. Однако по умолчанию ни один из этих параметров не будет выбран. Это поле со списком будет названо **ss**.

Для того чтобы открыть файл search.htm в отладчике скриптов Microsoft Script Debugger:

1. В меню кнопки **Start** выберите **Programs | Windows NT 4.0 Option Pack | Microsoft Script Debugger**, а затем укажите **Microsoft Script Debugger**.
2. Откройте страницу search.htm. Эта страница размещена в каталоге Microsoft Site Server\SiteServer\Knowledge\Search\database\search\FiveLakes\.

Для того чтобы поместить элемент combo box на поисковой странице, необходимо:

1. На поисковой странице search.htm поместить следующие строки кода:

```
<input type="text" name="qu" size="60" maxlength="100" value="">
</td>
</tr>
```

2. После тега `</tr>` поместить следующий код, формирующий элемент combo box (поле со списком):

```
<tr>
  <td><font face="verdana,arial,sans-serif"
    style="font-size: 10pt">
    Search for Publisher: </font>
  </td>
  <td>
    <select name=ss>
      <option value="">
      <option value="Microsoft Press">Microsoft Press
      <option value="Prentice Hall">Prentice Hall
    </select>
  </td>
</tr>
```

3. В меню **File** выбрать команду **Save** и затем закрыть данный файл. Итак, вы создали элемент combo box с именем `ss`, расположенный на поисковой странице. Далее вам необходимо будет добавить фрагмент кода, позволяющий производить расширенный поиск, на основе информации, введенной на странице. Этот программный код нужно внести в страницу результатов поиска results.asp в базе данных.

Для модификации страницы results.asp:

1. В отладчике Microsoft Script Debugger откройте страницу results.asp. Эта страница размещается в каталоге Microsoft Site Server\SiteServer\Knowledge \Search\database\search\FiveLakes\.

2. Поместите строку кода:

```
if Request("qu") <> "" then
  Response.write L_SearchingFor_text & " <b>" & _
  Request("qu") & "</b>. "
end if
```

В строку, находящуюся за оператором `end if`, введите следующий фрагмент кода, который изменит текст, появляющийся на итоговой странице (странице результатов):

```
if Request("ss") <> "" then
  Response.write L_SearchingFor_text & " <b>" & _
  Request("ss") & "</b>. "
end if
```

Теперь необходимо пояснить, как создается и выполняется объект запроса.

Выражение `set Q = Server.CreateObject("MSSearch.Query")` создает экземпляр объекта `Query`, используя `MSSearch.Query` в качестве программного идентификатора `(ProgID)` для объекта `Query`.

Свойства объекта `Query` должны устанавливаться на основе информации, вводимой пользователем, а также других критериев поиска. Строка `Q.SetQueryFromURL(Request.QueryString)` устанавливает параметр запроса. Выражение `Request.QueryString` осуществляет выборку информации, которую ввел пользователь на поисковой странице. Метод `SetQueryFromURL` устанавливает параметр данного запроса.

3. Поместите выражение:

```
Q.SetQueryFromURL(Request.QueryString)
```

В следующую строку внесите фрагмент кода, осуществляющий прием данных от элемента combo box с именем `ss`:

```
if request("ss") <> "" and request("qu") = "" then
  q.query="@meta_publisher= " & request("ss")
end if
if request("ss") <> "" and request("qu") <> "" Then
  q.query=q.query & " and @meta_publisher= " & _
  request("ss")
end if
```

Приведенный выше фрагмент кода осуществляет модификацию строки запроса, на основе данных, введенных пользователем на поисковой странице.

Вы можете изменить имя каталога, исправив свойство `Catalog` объекта `Query`. Строка `Q.Catalog = "FiveLakes"` указывает, что каталог, в котором осуществляется поиск, называется `FiveLakes`. вы можете добавить и другие каталоги, перечислив в списке их имена, разделенные запятой. Например, если вы хотите, чтобы поиск осуществлялся также и в каталоге `Hanson`, то данный фрагмент кода будет выглядеть `Q.Catalog = "FiveLakes,Hanson"`.

Каждый столбец в таблице должен быть определен до того, как он будет использоваться. Свойство `DefineColumn` определяет отображаемое имя столбца при помощи уникального идентификатора и имени столбца. Например, строка `Q.DefineColumn "sku = d1b5d3f0-c0b3-11cf-9a92-00a0c908dbf1 sku"` устанавливает отображаемое имя (display name) столбца `sku` как `sku`.

Объект `Query` имеет свойство `property`, которое позволяет вам определить, выборку каких столбцов из базы данных следует производить, а затем включать в результат поиска. Строка `Q.Columns = "DocTitle, DocAddress, Description, list_price, sale_start, sale_end, author, edition, category, new"` указывает, что столбцы `DocTitle`, `DocAddress`, `Description`, `list_price`, `sale_start`, `sale_end`, `author`, `edition`, `category` и `new` используются для выборки.

После выполнения запроса его результаты необходимо сохранить. Результаты поиска хранятся в наборе записей (recordset). Строка `set RS = Q.CreateRecordSet("sequential")` создает объект с именем `RS`, который хранит результаты поиска. Данный набор записей должен всегда проверяться, является ли он пустым или нет, а также для просмотра ошибок, которые могли возникнуть при выполнении запроса.

4. В меню **File** выберите команду **Save**, а затем закройте файл.

   Теперь вы можете выполнить поиск для того, чтобы посмотреть, что получилось в результате модификации файлов search.htm и results.asp.

Включение поисковых страниц в состав сайта

Каталог и поисковая страница, которые вы создали, недоступны для посетителей сайта. Поэтому необходимо создать ссылку на ваш сайт и, тем самым, предоставить пользователям доступ к поисковой странице.

На вашей странице, появляющейся по умолчанию, вам нужно организовать ссылку на поисковую страницу, добавив следующий фрагмент кода:

```
<A HREF="http://localhost/siteserver/knowledge/search/database/
    search/<sitename>/search.htm">
    To Jump to the Search page </A>
```

Лабораторная работа 2. Формирование каталога и поиск

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать каталог;
- носить модификации для обеспечения соответствующих поисковых возможностей.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь основные понятия об HTML и ASP-скриптах;
- быть знакомым с Microsoft Internet Explorer.

Примечание

Время, необходимое для выполнения данной работы, — 30 минут.

Примечание

Те студенты, которые уже выполнили все демонстрационные примеры и проанализировали программный код в данной главе, могут пропустить ту ее часть, в которой повторяются шаги по созданию каталога и модификации поиска.

Внимание!

Не следует использовать решения или же файлы с начальным кодом, представленные в данной лабораторной работе. Эти файлы приводятся лишь в качестве примера.

Упражнение 1. Формирование каталога

В этом упражнении вы создадите каталог базы данных на Web-сайте Ramona Publishing.

Для запуска Web-Based Administration (WebAdmin):

1. В меню кнопки **Start** выберите команду **Programs**.
2. Укажите пункт **Microsoft Site Server**, а затем команду **Administration**.
3. Выберите **Site Server Service Admin (HTML)**.

Для создания каталога базы данных при помощи WebAdmin:

1. В окне **Web Based Administration** нажмите кнопку **Search**.
2. В левом фрейме щелкните **Catalog Build Definition** (Определение для построения каталога).
3. В окне **Catalog Build Definition** (Определение для построения каталога) нажмите кнопку **Create**.
4. В окне **Create New Catalog Definitions Wizard** (Мастер создания определения нового каталога) укажите источник данных для каталога. В качестве источника данных вы будете использовать базу данных. Выберите **A database** (База данных) и нажмите кнопку **Next**.
5. В окне **Select Data Source and Specify Catalog Name** (Выбор источника данных и определение имени каталога) укажите базу данных и имя каталога. В поле **ODBC Data Source Name (DSN)** (Имя источника данных ODBC) выберите **Ramona**.
6. В поле **Catalog name** (Имя каталога) введите `Ramona`.
7. В поле **User name** (Имя пользователя) введите `sa`.
8. Поле пароля (password) оставьте пустым и нажмите кнопку **Next**.

   В окне **Select Table** (Выбор таблицы) выберите **Ramona_product** из раскрывающегося списка **Table** (Таблица), а затем щелкните по кнопке **Next**.
9. В окне **Select Content, Primary Key, and Hyperlink Columns** (Выбор столбцов содержания, первичного ключа и гиперссылок) в области **Content column** (Столбец содержимого) укажите столбец **description** (описание) базы данных.
10. В столбце **Primary key** (Первичный ключ) столбец **sku** задан по умолчанию. В области **Hyperlink column** (Столбец гиперссылок) выберите столбец **name** (имя), а затем нажмите кнопку **Next**.
11. Окно **Select Searchable and Retrievable Columns** (Выбор столбцов, по которым будет осуществляться поиск и выборка) позволяет вам определить те столбцы, которые в каталоге станут использоваться в качестве индексных при выборке и поиске. Воспользуйтесь табл. 2.6, чтобы указать столбцы, участвующие в поиске и выборке в базе данных. Установите соответствующие флажки и щелкните по кнопке **Next**.

Таблица 2.6. *Столбцы для поиска и выборки в базе данных*

Имя столбца	Используется при поиске	Используется при выборке
`sku`	Да	—
`list_price`	—	—

Таблица 2.6 (окончание)

Имя столбца	Используется при поиске	Используется при выборке
Image_file	—	—
Image_width	—	—
Image_height	—	—
sale_price	Да	Да
sale_start	Да	Да
sale_end	Да	Да
Author	Да	Да
Edition	Да	Да
Publisher	Да	Да
Category	Да	Да

12. В окне **Enable Immediate Build** (Разрешить формирование) щелкните **Build the database catalog now** (Сформировать каталог базы данных).
13. Нажмите кнопку **Finish** для начала формирования каталога.

    Доступ к странице поиска осуществляется по URL: **http://localhost /site-server/knowledge/search/database/search/Ramona/search.htm**.
14. Нажмите кнопку **OK**.

Упражнение 2. Модификация поисковой страницы

В данном упражнении вы произведете модификацию поисковых страниц Web-сайта Ramona Publishing для того, чтобы предоставить дистрибьюторам возможность осуществлять поиск новых книг.

Во-первых, необходимо поместить элемент combo box с именем `ss` на странице search.htm. Этот элемент должен иметь два значения: значение по умолчанию обязано быть пустым, а другое значение должно использоваться для отображения всех новых книг.

Затем, вам необходимо модифицировать файл Results.asp для того, чтобы изменить поисковый запрос.

Для того, чтобы открыть файл search.htm в отладчике скриптов Microsoft Script Debugger:

1. В меню кнопки **Start** выберите **Programs | Windows NT 4.0 Option Pack | Microsoft Script Debugger**, а затем укажите **Microsoft Script Debugger**.
2. Откройте страницу search.htm. Эта страница размещена в каталоге Microsoft Site Server\SiteServer\Knowledge\Search\database\search\Ramona\.

Для того чтобы расположить элемент combo box на поисковой странице, необходимо:

1. На поисковой странице search.htm поместить строки кода:

```
<input type="text" name="qu" size="60" maxlength="100" value="">
</td>
</tr>
```

2. После тега `</tr>` ввести следующий код, формирующий элемент combo box:

```
<tr>
  <td><font face="verdana,arial,sans-serif" style="font-size:10pt">
       Special search for: </font></td>
  <td>
    <select name=ss>
      <option value="">
      <option value="new">New Books
    </select>
  </td>
</tr>
```

3. В меню **File** выбрать команду **Save** и затем закрыть данный файл.

Для модификации файла results.asp:

1. Откройте страницу results.asp. Эта страница размещается в каталоге: Microsoft Site Server\SiteServer\Knowledge\Search\data-base\search\Ramona\.
2. Поместите строку кода:

```
if Request("qu") <> "" then
  Response.write L_SearchingFor_text & " <b>" & _
  Request("qu") & "</b>. "
end if
```

В строку, следующую за оператором `end if`, введите представленный ниже фрагмент кода:

```
if Request("ss") <> "" then
  Response.write L_SearchingFor_text & " <b>" & _
  Request("ss") & " " & "Books" & "</b>. "
end if
```

3. Поместите выражение:

```
Q.SetQueryFromUrl(Request.QueryString)
```

Добавьте код:

```
if request("ss") <> "" and request("qu") = "" then
  q.query="@meta_new= " & "new"
end if
if request("ss") <> "" and request("qu") <> "" Then
  q.query=q.query & " and @meta_new= " & request("ss")
end if
```

4. В меню **File** выберите команду **Save**, а затем закройте файл.

Для поиска новых книг:

1. В Internet Explorer откройте поисковую страницу сайта Ramona Publishing site. Доступ к поисковой странице осуществляется по URL: **http://localhost/siteserver/knowledge/search/database/search/Ra-mona/search.htm**.
2. В поле **Special search for** (Специальный поиск) выберите **New Books** (Новые книги) и затем нажмите кнопку **Search** для получения списка всех новых книг.

Вопросы для повторения

1. Объясните необходимость каталога на сайте электронной коммерции?
2. Каким образом поисковая страница включается в состав вашего сайта?
3. Какие требования предъявляются к определению каталога?

ГЛАВА 3

Создание и передача заказа

Краткий обзор

В электронной коммерции необходима безопасная и точная связь. Чтобы гарантировать целостность транзакций типа B2B, Microsoft Site Server 3.0 Commerce Edition предоставляет так называемые *конвейеры*[1] (pipelines), которые делают электронную связь более удобной.

В данной главе представлены конвейеры и показано, как их использовать.

Цели и задачи

После прочтения данной главы вы научитесь:

- ❒ объяснять такие понятия, как конвейер обработки заказа (Order Processing Pipelines) класса B2B, а также конвейер коммерческого обмена или CI-конвейер (CIP, Commerce Interchange Pipelines);
- ❒ описывать ступени в передающем элементе (Transmit pipeline element) CI-конвейера;
- ❒ создавать и передать заказ на закупку.

---

[1] В публикациях, посвященных вопросам электронной коммерции, можно встретить перевод этого термина как "соединение" (pipeline), под которым понимается "информация о правилах работы с документами внутри сервера. Соединяет между собой входящее соглашение с исходящим или листом распространения". — "КомпьютерПресс", № 9, 2000. В данном же случае будем придерживаться перевода этого термина как "конвейер", поскольку он отражает последовательную ступенчатую структуру, лежащую в основе этого понятия. Чтобы избежать многочисленных повторов громоздких наименований различных видов конвейеров, представляется целесообразным использовать однотипную аббревиатуру, например, "CI-конвейер", "CPS-конвейер" "CPP-конвейер" и т. д. В табл. 3.13 мы объединим сведения обо всех конвейерах. — *Пер.*

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod03\Mod03Code.txt.

OP-конвейеры

Когда две компании становятся электронными партнерами, они используют так называемые *конвейеры обработки заказа или OP-конвейеры* (OPP, Order Processing Pipelines) класса B2B для создания заказов на закупку и передачи их друг другу. Электронные партнеры применяют два типа конвейеров для создания и передачи заказов на закупку.

- *Конвейер плана корпоративной закупки* (CPP-конвейер, Corporate Purchasing Plan). Данный тип конвейера использует компоненты OP-конвейера для подсчета объема (итоговой стоимости) (total) заказа на закупку. На этот показатель влияют такие факторы, как скидки (discounts), налоги (taxes), а также транспортные расходы (shipping charges). Такой конвейер аналогичен конвейеру плана, P-конвейеру (Plan pipeline), используемому на сайтах электронной коммерции класса business-to-consumer.
- *Конвейер отправки корпоративной закупки* (CPS-конвейер, Corporate Purchasing Submit). Этот тип конвейера проверяет действительность заявки, передает и записывает заявки по заказам (order requisitions) на закупку в базу данных. Данный конвейер аналогичен конвейеру закупок, Pu-конвейеру (Purchase pipeline), используемому в сайтах электронной коммерции класса business-to-consumer.

CPP-конвейер

В электронной коммерции класса business-to-consumer обработка заказов производится на самом сайте. Поскольку такие виды деятельности, как формирование заказов на закупки и отправка их поставщикам, не входят в транзакции класса business-to-consumer (предприятие — заказчик), в этом случае используется конвейер плана, P-конвейер (Plan pipeline).

На рис. 3.1 отображено четырнадцать этапов, которые можно выделить в конвейере плана корпоративной закупки.

Однако в электронной коммерции класса B2B одна компания формирует заявку и передает ее другой фирме. По этому сценарию, CPP- и CPS-конвейеры работают в тандеме, обеспечивая формирование и проверку заказов на закупку. Следовательно, в транзакциях B2B CPP-конвейер применяется для вычисления итоговой стоимости заказа на закупку. В процессе этих вычислений данный конвейер учитывает все отчисления на рекламу, налоги, а также на транспортные и погрузочно-разгрузочные расходы.

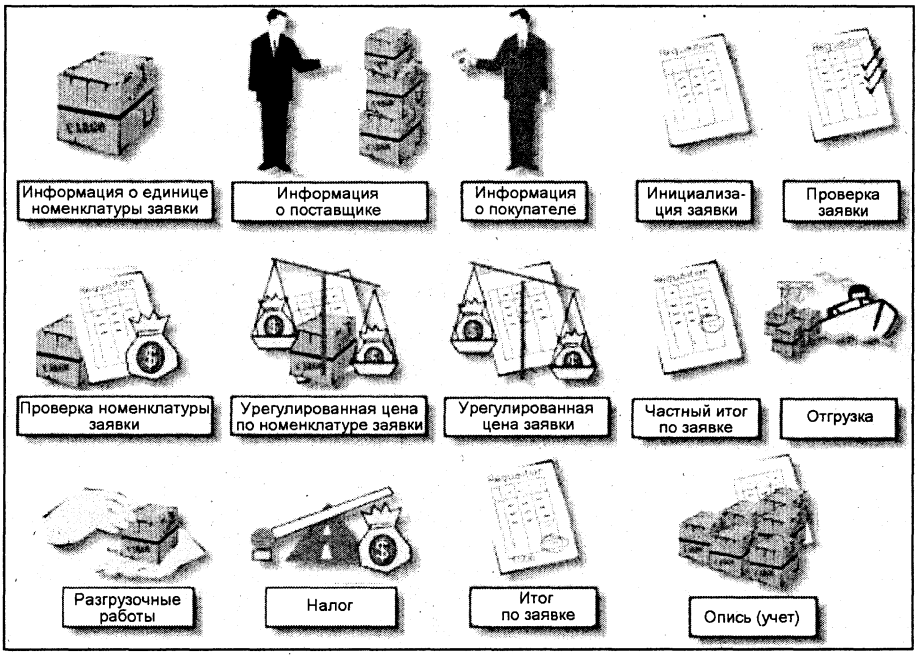


Рис. 3.1. Этапы конвейера плана корпоративной закупки (CPP-конвейер)

Сравнение CPP- и P-конвейеров

В CPP-конвейере насчитывается четырнадцать ступеней (stages), которые соотносятся с четырнадцатью ступенями P-конвейера, использующегося в коммерции класса business-to-consumer. CPP-конвейер создан при помощи шаблона CorpPurchasingPlan.pct. В табл. 3.1 приводится список ступеней CPP-конвейера, который распространяется также и на соответствующие ступени P-конвейера.

Таблица 3.1. *Ступени CPP- и P-конвейеров*

Ступени CPP-конвейера	**Соответствующие ступени P-конвейера, используемого в коммерции класса business-to consumer**
Информация о единице номенклатуры заявки (requisition item info)	Информация о продукции (product info)
Информация о поставщике (supplier information)	Информация о продавце (merchant information)
Информация о закупщике (buyer information)	Информация о покупателе (shopper information)

Таблица 3.1 (окончание)

Ступени CPP-конвейера	Соответствующие ступени P-конвейера, используемого в коммерции класса business-to consumer
Инициализация заявки (requisition initialization)	Инициализация заказа (order initialization)
Проверка заявки (requisition check)	Проверка заказа (order check)
Стоимость единицы номенклатуры заявки (requisition item price)	Стоимость единицы товара (item price)
Согласованная стоимость единицы номенклатуры заявки (requisition item adjust price)	Согласованная стоимость единицы товара (item adjust price)
Согласованная стоимость заявки (requisition adjust price)	Согласованная стоимость единицы номенклатуры заявки (requisition item adjust price)
Промежуточный итог заявки (requisition subtotal)	Промежуточный итог заказа (order subtotal)
Отгрузка, транспортировка (shipping)	Отгрузка, транспортировка (shipping)
Погрузочно-разгрузочные работы (handling)	Погрузочно-разгрузочные работы (handling)
Налог, пошлина (tax)	Налог, пошлина (tax)
Итог заявки (requisition total)	Итог заказа (order total)
Инвентаризация (inventory)	Инвентаризация (inventory)

CPS-конвейер

После успешной обработки формы заказа закупки, заказ можно передавать электронному партнеру. CPS-конвейер осуществляет проверку действительности заявки заказа на закупку, создает заказ на закупку и передает его электронному партнеру.

CPS-конвейер часто является конвейером для выполнения транзакций, T-конвейером (Transacted pipeline), в тех случаях, когда его компоненты осуществляют запись в базу данных. Конвейеры для выполнения транзакций состоят исключительно из компонентов, спроектированных и настроенных для поддержки транзакций сервера Microsoft Transaction Server (MTS).

Шаблоном для CPS-конвейера является `CorpPurchaseSubmit.pct`. Шаблон CPS-конвейера состоит из двух ступеней: ступени проверки заказа на закуп-

ку (Purchase Order Validate stage) и ступени передачи заказа на закупку[1] (Purchase Order Submit stage).

Ступень проверки заказа на закупку

Ступень проверки заказа на закупку осуществляет верификацию этого заказа.

Примечание

Site Server 3.0 Commerce Edition не содержит каких-либо компонентов для данной ступени. Однако вы можете включить в него[2] собственный компонент или же компонент третьей фирмы, осуществляющий проверку заказа.

Ступень передачи заказа на закупку

До момента передачи заказа на закупку, сам заказ должен быть создан и проверен. Данная ступень обеспечивает передачу подготовленного и проверенного заказа электронному партнеру.

В табл. 3.2 содержится подробная информация о компонентах, относящихся к данному этапу.

Таблица 3.2. *Компоненты ступени передачи заказа*

Компоненты	Описание
`ExecuteProcess`	Выполняет приложение на сервере с заданными аргументами
`MakePO`	Генерирует заказ на закупку на основе файла шаблона
`PipeToPipe Transfer`	Осуществляет передачу формы заказа (OrderForm) или словаря (Dictionary) из одного работающего конвейера в другой
`POtoFile`	Посылает заказ на закупку (обычно, результат MakePO) в файл
`SaveReceipt`	Записывает содержимое, введенное в форму заказа (OrderForm) в память квитанций (receipt storage)
`SendSMTP`	Посылает почтовое сообщение по указанному адресу
`SQLItem`	Запускает указанную SQL-команду для каждого элемента в заказе на закупку. Этот компонент использует поля из заказа и элементы в качестве своих аргументов

[1] Понятие ступеней более подробно раскрыто далее в *разд. "Архитектура CI-конвейера и транспортный словарь"* этой главы — *Пер.*

[2] То есть в Site Server 3.0 Commerce Edition. — *Ред.*

Таблица 3.2 (окончание)

Компоненты	Описание
SQLItemADO	Запускает указанную SQL-команду для каждого элемента в заказе на закупку и работает с полями из заказа и элементами в качестве своих аргументов. Этот компонент аналогичен компоненту SQLItem, за исключением того, что он использует Microsoft ActiveX Data Objects (ADO) и может быть включен в Т-конвейер
SQLOrder	Запускает указанную SQL-команду (один раз на заказ по поставкам), используя поля из заказа в качестве аргументов
SQLOrderADO	Запускает указанную SQL-команду (один раз на заказ по поставкам), используя поля из заказа в качестве аргументов. Этот компонент аналогичен компоненту SQLOrder за исключением того, что он работает с ADO и может быть включен в конвейер для выполнения транзакции (Т-конвейер)

В следующем разделе вы узнаете, как использовать компоненты MakePO и POtoFile, относящиеся к ступени отправки заказа на закупку.

Генерация заказа на закупку

Компонент MakePO применяется для генерации заказа на закупку (PO, Purchase Order) при помощи шаблона заказа, выбирая для этого информацию из формы заказа на закупку. Компонент MakePO имеет установки (параметры), перечисленные в табл. 3.3.

***Таблица 3.3.** Компоненты ступени генерации заказа*

Установка	Описание
Template File Name (имя файла шаблона)	Указывает имя файла, содержащего шаблон заказа на закупку. Этим файлом обычно является файл Microsoft Visual Basic Scripting (VBS) с расширением txt
Script language for the template (скрипт-язык шаблона)	Указывает язык программирования, используемый в данном шаблоне. По умолчанию — VBScript
Output Property Name (имя выводимого свойства)	Указывает имя поля на форме заказа, в которое данный компонент записывает PO. По умолчанию — _po_text

Таблица 3.3 (окончание)

Установка	Описание
Child Object Name (имя дочернего объекта)	Указывает объект, на который есть ссылка в транспортном словаре (Transport Dictionary) по паре "имя/значение", передаваемой в конвейере. Этот компонент применим лишь в том случае, если выбрана опция **Use Child Object** (Использовать дочерний объект)
Use Child Object (использовать дочерний объект)	Указывает источник, из которого данный компонент считывает информацию заказа на закупку. Выбирайте эту опцию, если используете CI-конвейер для того, чтобы данный компонент считывал эту информацию из объекта бизнес-данных (business data object), указанного в поле **Child Object Name**

Запись заказа на закупку в файл

Для записи заказа на закупку в файл используется компонент `POtoFile`. У этого компонента имеются установки (параметры), приведенные в табл. 3.4.

***Таблица 3.4.** Параметры компонента* `PotoFile`

Установка	Описание
Source Field Name (имя поля источника)	Указывает имя поля на форме заказа, из которого компонент считывает данные заказа на закупку
File Name (имя файла)	Указывает путь к файлу, в который записан заказ на закупку
File named in field (файл, указанный в поле)	Указывает имя поля на форме заказа, значением которого является имя файла, предназначенное для осуществления в него записи заказа на закупку
Temporary file, name saved in field (временный файл, имя сохраняемое в поле)	Указывает, что данному компоненту необходимо сохранить заказ на закупку во временном файле, а затем сохранить имя временного файла в поле, которое здесь указано

CI-конвейеры

В предыдущем разделе вы узнали о так называемых конвейерах обработки заказа — OP-конвейерах. Эти конвейеры используются для вычисления итоговой стоимости заказа на закупку, проверяют заявку (requisition) заказа и окончательно формируют заказ. В электронной коммерции класса B2B

сформированный заказ на закупку должен быть передан поставщику. Сервер Site Server 3.0 Commerce Edition предоставляет ряд так называемых конвейеров коммерческого обмена, CI-конвейеров (Commerce Interchange Pipelines), которые вы можете использовать для обмена информацией с электронными партнерами. В следующих разделах вы познакомитесь с этими конвейерами, их архитектурой и применением для обмена документами с электронными партнерами.

Краткий обзор

CI-конвейер предоставляет возможность для компаний любого масштаба обмениваться информацией в электронном виде. Электронные партнеры используют данный тип конвейера для безопасной передачи заказов на закупку, квитанций, нарядов на отгрузку, счетов-фактур, записей счетов (billing records) и т. д. Взаимодействие может осуществляться на одиночном компьютере, между компьютерами, соединенными в локальной сети (LAN), глобальной сети (WAN), частной сети (VAN, Value Added Network) или Internet.

На рис. 3.2 представлены операции CI-конвейера.

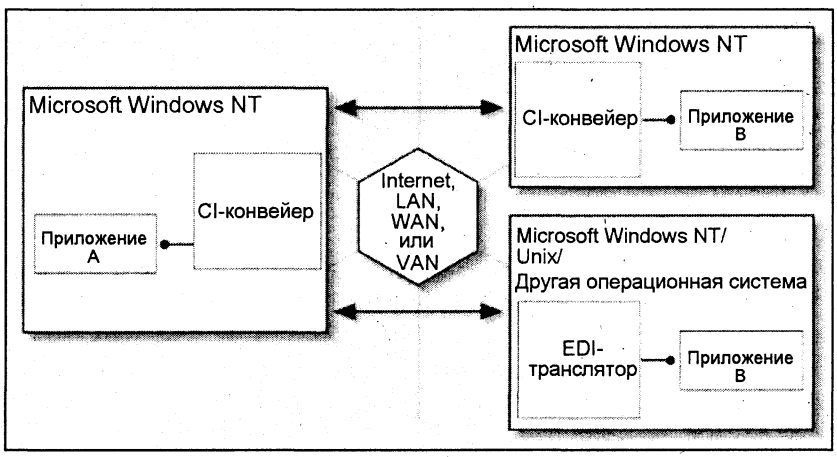


Рис. 3.2. Операции CI-конвейера

Достоинства CI-конвейера

CI-конвейер представляет собой наиболее целесообразное решение для электронных транзакций по следующим причинам:

- CI-конвейер может быть сконфигурирован так, чтобы обеспечить как передачу, так и прием бизнес-объектов;

- CI-конвейер поддерживает различные сценарии торговли и взаимодействия в коммерции класса B2B, включая корпоративные закупки (corporate purchasing), состояние заказа (order status);
- CI-конвейер взаимодействует с существующими транспортными системами, такими как e-mail, HTTP, а также с новыми системами — Distributed Component Object Model (DCOM) и Microsoft Message Queue (MSMQ);
- CI-конвейер не зависит ни от формата, ни от "транспорта" данных. Таким образом, разработчики и поставщики сторонних фирм могут создавать совместимые компоненты, которые можно связывать в различной конфигурации по желанию заказчика.

Преимущества CI-конвейера перед OP-конвейером

Как CI-конвейеры, так и OP-конвейеры могут использоваться для передачи заказов на закупку. Однако у CI-конвейеров имеется ряд функциональных возможностей, которые дают ему некоторые преимущества перед OP-конвейерами (табл. 3.5).

Таблица 3.5. *Сравнение возможностей CI- и OP-конвейеров*

CI-конвейеры	OP-конвейеры
Компоненты для преобразования (mapping components) могут использоваться для конвертирования объектов данных (data objects) в поток байтов, пригодный для передачи	Эта возможность в OP-конвейере отсутствует
Могут передаваться компоненты `SendSMTP`, `SendHTTP`, `SendDCOM`, а также компоненты третьих фирм	Поддерживается только `SendSMTP`
Информация может передаваться в зашифрованном виде	Шифрование не поддерживается

Архитектура CI-конвейера и транспортный словарь

В транзакциях типа B2B некоторая компания генерирует заказ на закупку и затем передает его своему поставщику. Для передачи и приема таких заказов используется CI-конвейер.

На рис. 3.3 представлено взаимодействие CI-конвейера и транспортного словаря (Transport dictionary).

Прежде, чем можно будет передать или принять заказ на закупку, CI-конвейеры должны будут выполнить такие операции, как шифрование/дешиф-

рование, цифровая подпись и т. д. Поэтому объекты бизнес-данных, хранящиеся в заказе на закупку, конвертируются в так называемый транспортный словарь (Transport dictionary). Каждый компонент CI-конвейера определенным образом влияет на объект словаря.

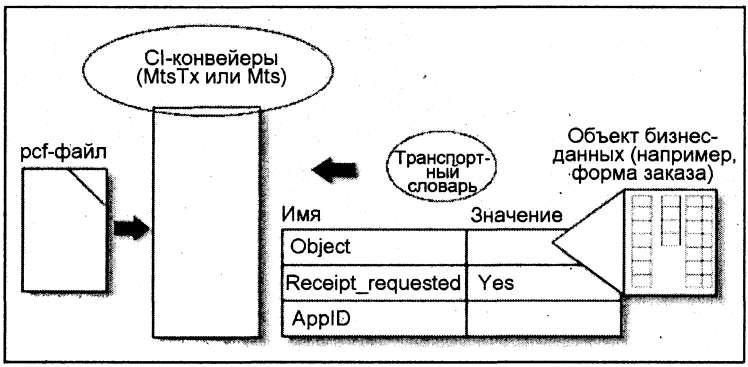


Рис. 3.3. Взаимодействие CI-конвейера и транспортного словаря

Архитектура CI-конвейера

CI-конвейер является реализацией архитектуры конвейера на сервере Site Server 3.0 Commerce Edition. Каждый конвейер представляет собой некую "конструкцию" (framework), последовательно выполняющую процессы. Эта "конструкция" состоит из ступеней (stages), описывающих разновидность работы, выполняемой одним или более компонентами, относящимися к данной ступени. Каждая ступень может быть полностью адаптирована к вашему приложению.

Любая ступень CI-конвейера состоит из одного или более компонентов. Компонент представляет собой действующий COM-сервер (COM, Component Object Model — модель компонентных объектов), выполняющий некоторую операцию над объектом бизнес-данных. Например, ступень шифрования CI-конвейера (CIP Encryption stage) может содержать компонент `EncryptPCKS`, предназначенный для шифрования объекта бизнес-данных. На определенной ступени некий компонент принимает данные от предыдущего компонента, выполняет свой процесс над объектом данных и затем предоставляет результаты следующему компоненту. Вы можете настраивать приложение электронной коммерции, добавляя, удаляя или же конфигурируя компоненты конвейера.

Транспортный словарь

Транспортный словарь (Transport dictionary) — это объект, который приложения передают в CI-конвейер, и он является целью всей деятельности CI-конвейера.

Приложения создают и передают транспортный словарь в качестве аргумента в метод `Execute`, принадлежащий CI-конвейеру. При вызове метода `Execute` компоненты CI-конвейера считывают данные из транспортного словаря в память, выполняют операции с данными, а затем записывают результаты обратно в транспортный словарь.

Типы CI-конвейеров

В типичном примере взаимодействия B2B используются два CI-конвейера, запущенных на разных серверах:

- конвейер для передачи, CIT-конвейер (Transmit pipeline). Передает по сети, от одного торгового партнера к другому партнеру, объект бизнес-данных;
- конвейер для приема, CIR-конвейер (Receive pipeline). Принимает объект бизнес-данных, распаковывает этот объект и включает его в состав приложения.

На рис. 3.4 показаны ступени при типичном взаимодействии B2B.

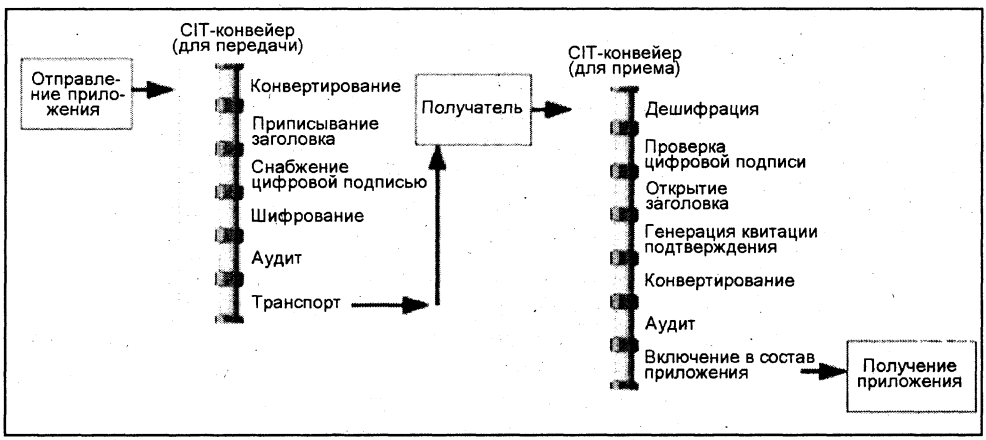


Рис. 3.4. Ступени при типичном взаимодействии B2B

CI-конвейер также обеспечивает подтверждение полного прохождения документа. В сочетании с цифровыми подписями, квитанции подтверждения (return receipts) обеспечивают верификацию бизнес-взаимодействия по Internet.

Конфигурация отдельного конвейера сохраняется в файле конфигурации конвейера (pcf, pipeline configuration file). В табл. 3.6 представлена подробная информация о типах компонентов, содержащихся в конфигурации конвейера.

Таблица 3.6. *Типы компонентов в конфигурации CI-конвейера*

Типы компонентов	Описание
Конвертирующие компоненты (Mapping components)	Преобразуют либо содержимое объектов бизнес-данных в формат, предназначенный для передачи данных, либо полученные данные в объект, в котором эти данные изначально хранились
Компоненты приписывания/открытия заголовка (Add/Open Header components)	Добавляют/удаляют элементы в/из транспортный словарь, включая адреса отправителя и получателя, а также квитанции подтверждения
Компоненты цифровой подписи (Digital Signature components)	Определяют тип цифровой подписи (если предусматривается) с целью заверения ею объекта бизнес-данных или же для верификации подписи объекта
Компоненты шифрования/дешифрования (Encryption/Decryption components)	Определяют, как шифровать или дешифровать содержимое объекта бизнес-данных, используя сертификаты
Компоненты аудита (Audit components)	Записывают различные фрагменты объекта бизнес-данных в базу данных аудита[1]
Транспортный коннектор (Transport connector)	Разработан и настроен для передачи или приема объекта бизнес-данных
Включение в состав приложения (Application Integration)	Вызывает приложение для обработки полученного объекта бизнес-данных

Для каждой ступени и компонента в конвейере, служащем для передачи (CIT-конвейер), есть свой аналог в том конвейере, который предназначен для приема (CIR-конвейер). В конвейере для приема имеется дополнительный компонент, именуемый генерацией (формированием) квитанции (generate receipt).

Создание и передача заказа на закупку

В транзакции типа B2B, CIT-конвейер используется для упаковки объектов бизнес-данных и передачи их электронным партнерам. Объект бизнес данных проходит шесть ступеней в этом конвейере, прежде чем будет передан электронному партнеру. Шесть ступеней подробно рассмотрены в следующем разделе.

[1] То есть контроля полномочий. — *Пер.*

Ступени CIT-конвейера

CIT-конвейер является разновидностью CI-конвейера, который упаковывает объекты бизнес-данных и передает их электронным партнерам. Шаблоном для этого конвейера является файл Transmit.pct. В данном конвейере имеется шесть ступеней:

- конвертирование (Map stage);
- приписывание заголовка;
- добавление цифровой подписи;
- шифрование;
- аудит;
- транспортировка.

На рис. 3.5 представлены перечисленные ступени.

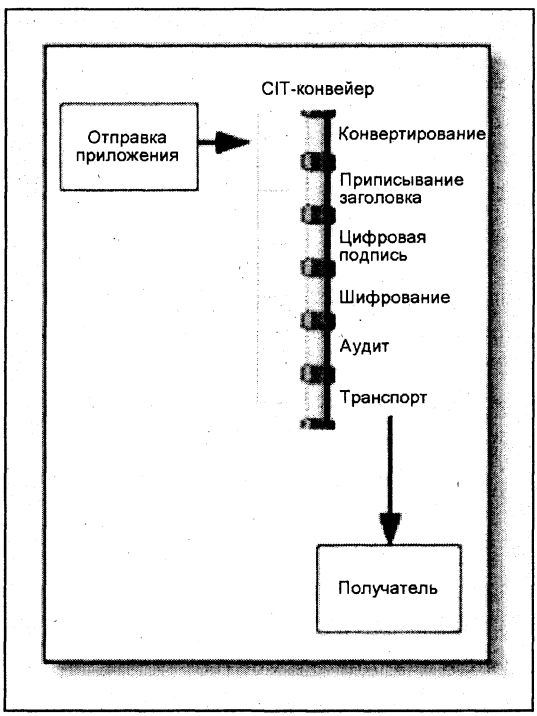


Рис. 3.5. Ступени CIT-конвейера

Конвертирование объекта бизнес-данных

Конвертирование описывает процесс преобразования данных объекта из одного формата в другой. Ступень конвертирования может включать в себя

компоненты третьих фирм, предназначенные для преобразования EDI[1]-данных в форму, подходящую для их передачи.

В CIT-конвейере ступень преобразования может содержать компонент MapToXML. Этот компонент считывает данные из объекта, указанного парным параметром "имя/значение", конвертирует их либо в заключенный между тегами XML поток двоичных данных, либо в поток XML-данных, и записывает результаты в транспортный словарь в виде новых парных параметров, именуемых working_data (рабочие данные).

Давайте рассмотрим пример следующего объекта словаря (dictionary object):

```
Set myObj = Server.CreateObject("Commerce.Dictionary")
myObj.Company = "Microsoft"
```

Когда этот объект словаря передается в компонент MapToXML, данный компонент генерирует следующие данные:

```
<MAPXML><VALUE dt:dt="object" dt:classid="classid:
  B7990D09-45FD-11D0-8176-00A0C90A90C7">
  <DICTIONARY>
    <DICTITEM key="Company"><VALUE dt:dt="string" xml-space="preserve">
    Microsoft</VALUE></DICTITEM>
  </DICTIONARY>
</VALUE></MAPXML>
```

Приписывание заголовка

Ступень приписывания заголовка (add header) инкапсулирует рабочие данные в заголовок, который несет дополнительную информацию о передаче и записывает результат снова в область рабочих данных.

Когда вы передаете объект бизнес-данных с помощью CIT-конвейера, вы можете запросить подтверждение передачи данных. Для этого в транспортный словарь включается свойство Receipt_requested, со значением Yes.

```
TransportDictionary.Receipt_requested= "Yes"
```

Если в данной ступени CI-конвертора имеется свойство Receipt_requested, и его значение равно Yes, то перед началом передачи компонент AddHeader записывает запрос подтверждения в транспортный словарь.

Запрос подтверждения включает в себя следующие элементы:

- алгоритм хэширования данных бизнес-объекта для получения комбинированного сообщения. Стандартным хэш-алгоритмом является SHA1;

---

[1] EDI — Electronic Data Interchange, электронный обмен данными (деловыми документами) между компьютерными программами различных компаний в стандартизированной форме. — *Пер.*

- тег типа документа, который идентифицирует вид документа, запрошенного в конвейере для приема. Его значение всегда равно `Receipt`, указывая на наличие запроса подтверждения.

Подробная информация о компонентах, относящихся к данной ступени, приведена в табл. 3.7.

Таблица 3.7. *Компоненты ступени приписывания заголовка*

Компонент	Описание
`AddHeader`	Добавляет серию XML-тегов к рабочим данным, включая идентификатор транзакции, дату и время транзакции и, в качестве опции, запрос на подтверждение приема, а также адреса отправителя и полу́чателя
`EncodeMIME`	Осуществляет MIME-кодирование рабочих данных типа "имя/значение"

Добавление цифровой подписи данных

Транспортный словарь после ступени приписывания заголовка содержит все необходимые данные для осуществления передачи. Чтобы обеспечить секретность передачи, вы можете снабдить данные цифровой подписью.

Ступень цифровой подписи обеспечивает размещение цифровой подписи в потоке байтов. В необязательном порядке, вы можете использовать такие компоненты в этой ступени для S/MIME-кодирования данных.

К данной ступени относятся компоненты, перечисленные в табл. 3.8.

Таблица 3.8. *Компоненты ступени добавления цифровой подписи*

Компонент	Описание
`DigitalSig`	Обеспечивает цифровую подпись в парах "имя/значение" для рабочих данных, используя Public Key Crypto System 7 (PKCS7)
`EncodeSMIME`	Обеспечивает MIME-кодирование данных в парах "имя/значение"

Шифрование данных

Ступень шифрования обеспечивает шифрование передаваемых данных таким образом, чтобы они не были видны тем пользователям, которые не имеют соответствующих полномочий. В этих компонентах применяется технология шифрования, которая поддерживается системой Microsoft Cryptographic Application Interface (CryptoAPI) и включает в себя:

- поддержку стандарта Public Key Crypto System (PKCS7);
- S/MIME-инкапсуляцию EDI Messages (сообщений EDI) с использованием проекта стандарта Internet IETF EDIINT;
- компоненты третьих фирм.

Описание компонентов данной ступени приведено в табл. 3.9.

Таблица 3.9. *Компоненты ступени шифрования данных*

Компонент	Описание
EncryptPKCS	Шифрует объект бизнес-данных в соответствии со стандартом Public Key Crypto System (PKCS)
EncodeSMIME	Использует стандарт безопасного MIME-кодирования по отношении к парам "имя/значение" рабочих данных. Эта операция может включать в себя цифровую подпись и шифрование объекта бизнес-данных

Аудит бизнес-транзакции

Ступень аудита записывает выбранные элементы бизнес-транзакции в базу данных аудита.

К этой ступени относятся компоненты, представленные в табл. 3.10.

Таблица 3.10. *Компоненты ступени аудита*

Компонент	Описание
Audit	Записывает указанные значения из транспортного словаря в таблицу базы данных аудита
AuditReceipt	Записывает данные из квитанции в таблицу базы данных аудита. Этот компонент не используется в CIT-конвейере

Транспортирование данных

Транспортная ступень содержит компоненты, которые подготавливают рабочие данные для передачи, а затем передают эти данные. Рассматриваемая ступень берет на себя все функции, связанные с транспортным протоколом.

В настоящее время поддерживаются транспортные протоколы третьих фирм — DCOM, SMTP и HTTP.

К этой ступени относятся компоненты, перечисленные в табл. 3.11.

Таблица 3.11. *Компоненты транспортной ступени*

Компоненты	Описание
`PipetoPipeTransfer`	Инициализирует и обеспечивает работу удаленного конвейера (remote pipeline) для получения транспортного словаря из текущего конвейера
`SendSMTP`	Конвертирует снабженные подписью и зашифрованные рабочие данные в текст, и затем посылает их как сообщение электронной почты к указанному получателю
`SendHTTP`	Конвертирует снабженные подписью и зашифрованные рабочие данные в текст, и затем посылает их с помощью метода POST в HTTP на Web-страницу
`SendDCOM`	Посылает преобразованные рабочие данные при помощи Distributed Component Object Model (DCOM) для создания и выполнения MtsTxPipeline или MtsPipeline на удаленном компьютере с Commerce Server

Шаги создания и передачи заказа на закупку

Когда какая-нибудь компания желает сделать заказ, с помощью транзакции в электронной коммерции, она должно спланировать, а затем передать этот заказ своим электронным партнерам. В данном разделе вы узнаете о том, как создавать и передавать электронным способом заказ на закупку.

Рассмотрим сценарий электронной коммерции, по которому Web-сайт FiveLakes генерирует заказ на закупку, а затем отправляет его на Web-сайт Ramona Publishing. Для создания и передачи данного заказа компания "FiveLakes" должна выполнить следующие шаги.

- Создать и активизировать CPP-конвейер.

  Когда вы запускаете Site Builder Wizard (Мастер построения сайта) сайта электронной коммерции, то автоматически создаются ASP-страницы формы заказа i_util.asp и xt_orderform_purchase.asp. Эти файлы служат для создания и активизации CPP-конвейера. Вы должны модифицировать указанные файлы для того, чтобы создать и активизировать CPP-конвейер.

- Создать CPS-конвейер.

  Необходимо создать этот конвейер для отправки заказа на закупку. Используются следующие компоненты конвейера и их функции:

  - `MakePO`. Генерирует заказ на закупку.
  - `POtoFile`. Записывает заказ на закупку в файл.

- `Scriptor`. Выполняет следующие задачи: создает Т-конвейер (Transacted pipeline) и загружает файл конфигурации конвейера; создает транспортный словарь; активизирует метод `Execute` данного конвейера.

❒ Модифицировать код для активизации CPS-конвейера.

Нужно изменить ASP-страницу xt_orderform_purchase.asp для активизации CPS-конвейера.

❒ Создать CIT-конвейер.

Необходимо создать CIT-конвейер для передачи заказа на закупку поставщику.

Активизация CPP-конвейера на странице i_util.asp

CPP-конвейер используется для вычисления итоговой стоимости заказа на закупку (purchase order total), включая все ценовые скидки для стимулирования сбыта и увеличения объема продаж (promotional discounts), налоги, а также затраты на транспортирование и погрузочно-разгрузочные работы. Вам необходимо модифицировать функцию `UtilRunPlan` на странице i_util.asp для того, чтобы активизировать новый CPP-конвейер. Существующий код для активизации CPP-конвейера в функции `UtilRunPlan` выглядит так:

```
errorLevel = UtilRunPipe("plan.pcf", mscsOrderForm, mscsPipeContext)
```

Его необходимо преобразовать:

```
errorLevel = UtilRunPipe("CorpPlan.pcf", mscsOrderForm, mscsPipeContext)
```

В данном фрагменте кода CorpPlan.pcf является именем CPP-конвейера.

Активизация CPP-конвейера на странице xt_orderform_purchase.asp

На странице xt_orderform_purchase.asp функция `OrderFormPurchase` вызывает функцию `UtilRunPipe` со страницы i_util.asp для загрузки и выполнения Р-конвейера. Существующий программный код для активизации этого конвейера в функции `OrderFormPurchase` выглядит следующим образом:

```
errorLevel = UtilRunPipe("plan.pcf", mscsOrderForm, mscsPipeContext)
```

Данный код изменен для того, чтобы активизировать CPP-конвейер:

```
errorLevel = UtilRunPipe("CorpPlan.pcf", mscsOrderForm, mscsPipeContext)
```

Этот фрагмент кода запускает указанные ступени в конвейере, которые воздействуют на форму `OrderForm`.

Создание CPS-конвейера

Вы создаете CPS-конвейер для генерации заказа на закупку. В ступень передачи заказа на покупку (Purchase Order Submit stage) вы включаете компо-

нент MakePO для генерации заказа путем слияния информации, содержащейся в форме заказа, с информацией шаблона заказа на закупку.

Компонент *MakePO*

Подобно ASP-странице шаблон MakePO содержит простой текст, чередующийся специальными тегами, в которых содержится скрипт-код. Вывод информации из этого скрипта может осуществляться при помощи выражения PageGen.Print или конструкции <%% = выражение %%>.

Внутри шаблона MakePO бизнес-объект обрабатывается как объект с именем Item. Эту задачу выполняет следующий код:

```
<%%
  Dim nItemCount
  nItemCount = Items.Items.count
  If (nItemCount > 0) Then
    Dim iItem,Item
    For iItem = 0 To nItemCount - 1
      Set Item = Items.Items.Item(iItem)
%%>
      Order ID : <%% =Items.order_id %%> Item
      Number:<%%= Item.sku %%> Qty:<%%= Item.quantity %%>
<%%
    Next
  End If
%%>
```

Компонент *POtoFile*

Формируемые вами заказы на закупку могут записываться в файл при помощи компонента POtoFile. Местоположение данного файла должно указываться в свойствах этого компонента.

Компонент *Scriptor*

Scriptor является завершающим компонентом, выполняемым в CPS-конвейере. Он реализует следующие действия:

- создает Т-конвейер и загружает файл конфигурации конвейера;
- создает транспортный словарь;
- активизирует метод Execute конвейера.

Компонент Scriptor создает и обеспечивает выполнение CIT-конвейера. Данная функция принимает в этом скрипте четыре аргумента:

```
Function mscsexecute(config, orderform, context, flags)
```

Создание Т-конвейера и загрузка его в конфигурационный файл

Эта функция начинается с создания Т-конвейера (mscsMtsTxPipeline) и затем загружает файл конфигурации конвейера Transmit.pcf.

```
Dim mscsMtsTxPipeline
Set mscsMtsTxPipeline = CreateObject("Commerce.MtsTxPipeline")
Call mscsMtsTxPipeline.loadpipe("transmit.pcf")
```

Формирование транспортного словаря

Далее, эта функция создает транспортный словарь для его передачи в качестве аргумента в CI-конвейер. Транспортный словарь содержит парный параметр "имя/значение", который хранит данные об отправляемом объекте бизнес-данных.

```
Dim TransportDictionary
Set TransportDictionary = CreateObject("Commerce.Dictionary")
```

Значение с именем Object включается в транспортный словарь и указывается в форме заказа следующим образом:

```
Set TransportDictionary.object = orderform
```

Свойство object используется первым компонентом CIT-конвейера. Компонент MapToXML, который считывает object, преобразует его в двоичную последовательность и записывает результат обратно — в свойство working_data.

В том случае, если запрашивается квитанция подтверждения (получения информации), необходимо добавить следующую строку:

```
TransportDictionary.Receipt_requested = "Yes"
```

Активизация метода *Execute* конвейера

Наконец, данная функция активизирует метод Execute конвейера, передавая транспортный словарь и словарь PipeContext в качестве аргументов следующим образом:

```
errorlevel = mscsMtsTxPipeline.Execute(1, _
TransportDictionary, context, 0)
```

Активизация CPS-конвейера на странице xt_orderform_purchase.asp

CPS-конвейер необходимо активизировать на странице xt_orderform_purchase.asp. Он должен быть исполнен после подтверждения закупки и до уничтожения формы заказа.

На странице xt_orderform_purchase.asp функция `OrderFormPurchase` вызывает функцию `UtilRunTxPipe` для загрузки и выполнения CPS-конвейера в режиме транзакции.

```
errorLevel = UtilRunTxPipe ("CorpSubmit.pcf", mscsOrderForm, _
mscsPipeContext)
```

Создание CIT-конвейера

CIT-конвейер начинается с объекта, который передается в него в транспортном словаре. В то время когда конвейер работает, различные компоненты записывают новые значения в транспортный словарь.

Теперь вы создадите CIT-конвейер для передачи заказа с Web-сайта FiveLakes на сайт Ramona Publishing. Ступени данного конвейера будут содержать в себе компоненты, перечисленные в табл. 3.12.

Таблица 3.12. *Компоненты ступеней CIT-конвейера*

Ступени CIT-конвейера	Необходимые компоненты
Конвертирование	`MapToXML`
Приписывание заголовка	`AddHeader`
Добавление цифровой подписи	`DigitalSig`
Шифрование	`EncryptPKCS`
Аудит	`Audit`
Транспортировка	`SendHTTP`

В ступени транспортировки компонент `SendHTTP` служит для отправки данных бизнес-объекта в качестве аргумента по запросу HTTP. Компонент `SendHTTP` использует команду POST в HTTP для передачи данных из указанного поля транспортного словаря или формы заказа на Web-страницу, URL которой задан в поле данного компонента.

```
Localhost/Ramona/getdata.asp
```

Примечание

Компоненты конвейера используются именно для этого сценария. Вы можете добавлять, удалять или модифицировать их, исходя из потребностей своей компании.

Когда конвейер завершит свою работу, транспортный словарь автоматически уничтожится, а объект исходной формы заказа останется.

Перечень конвейеров

Для того чтобы объединить все изложенные сведения о конвейерах, приведем сводную табл. 3.13[1].

Таблица 3.13. *Перечень конвейеров, упомянутых в данной книге*

Условное наименование конвейера	Оригинальное название конвейера	Полный перевод наименования конвейера	Класс конвейера	Основное назначение конвейера
CI-конвейер	CIP, Commerce Interchange Pipelines	Конвейер коммерческого обмена	B2B	Конвейер коммерческого обмена сервера Site Server 3.0 Commerce Edition для обмена информацией с электронными партнерами
CIT-конвейер	Transmit pipeline	Конвейер для передачи	B2B	Передает по сети, от одного торгового партнера к другому партнеру, объект бизнес-данных
CIR-конвейер	Receive pipeline	Конвейер для приема	B2B	Принимает объект бизнес-данных, распаковывает этот объект и включает его в состав приложения
OP-конвейер	OPP, Order Processing Pipelines	Конвейер обработки заказа	B2B	Создает заказы на закупку и передает их электронному партнеру
CPP-конвейер	Corporate Purchasing Plan	Конвейер плана корпоративной закупки	B2B	Используется для подсчета объема (итоговой стоимости) заказа на закупку. Этот конвейер аналогичен конвейеру плана, P-конвейеру, используемому на сайтах электронной коммерции класса business-to-consumer (B2C)
CPS-конвейер	Corporate Purchasing Submit	Конвейер отправки корпоративной закупки	B2B	Проверяет действительность заявки, передает и записывает заявки по заказам на закупку в базу данных. Данный конвейер аналогичен конвейеру закупок, Pu-конвейеру, используемому в сайтах электронной коммерции класса B2C

[1] Данная таблица объединяет сведения о конвейерах, представленных в этой главе и используемых далее, и является добавлением к тексту оригинала. — *Пер.*

Таблица 3.13 (окончание)

Условное наименование конвейера	Оригинальное название конвейера	Полный перевод наименования конвейера	Класс конвейера	Основное назначение конвейера
P-конвейер	Plan pipeline	Конвейер плана	B2C	Обработка заказов на сайте. Аналогичен CPP-конвейеру B2B
Pu-конвейер	Purchase pipeline	Конвейер закупок	B2C	Аналогичен CPS-конвейеру B2B
T-конвейер	Transacted pipeline	Конвейер для транзакций		1. Конвейер, служащий для выполнения транзакций. 2. Объект-конвейер mscsMtsTxPipeline. Используется для загрузки CIT- и CIR-конвейеров, для отправки квитанций подтверждения. Может включаться в состав приложений

Лабораторная работа 3. Формирование и передача заказа на закупку

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать CPP-конвейер;
- формировать CPS-конвейер;
- создавать CIT-конвейер;
- осуществлять передачу заказа на закупку с помощью CIT-конвейера.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь основные понятия об HTML и ASP-скриптах;
- быть знакомым с Microsoft Internet Explorer и Microsoft Visual InterDev 6.0.

Подготовка к лабораторной работе

Для выполнения данной работы вы должны скопировать файлы из папки <папка установки>\Labs\Lab03\FiveLakes\StartCode в папку C:\Inetpub\wwwroot\Fivelakes. Нажмите кнопку **Yes**, если поступит предложение заменить новыми существующие файлы и папки.

Примечание

Время, необходимое для выполнения данной работы, — 45 минут.

Упражнение 1. Создание и запуск CPP-конвейера

В этом упражнении мы создадим CPP-конвейер и напишем код для его активизации на Web-сайте FiveLikes, созданном в качестве примера.

Для запуска редактора конвейера нажмите кнопку **Start**, выберите пункт **Programs** и, далее, **Microsoft Site Server** и **Commerce**, а затем укажите команду **Pipeline Editor**.

Для того чтобы изменить P-конвейер на CPP-конвейер на сайте FiveLakes:

1. В меню **File** выберите команду **New**. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) отобразится список имеющихся шаблонов.
2. Выберите файл CorpPurchasingPlan.pct и затем нажмите кнопку **OK**.
3. В меню **File** выберите команду **Open**.
4. Под папкой C:\InetPub\wwwroot\FiveLakes выберите папку Config.
5. В папке Config укажите файл plan.pcf, и затем нажмите кнопку **Open** для того, чтобы открыть существующий P-конвейер.
6. В меню **Window** выберите команду **Tile** (Упорядочить без перекрытия). Два окна будут упорядочены горизонтально.
7. Под **Product Info** в P-конвейере щелкните правой кнопкой мыши на элементе **QueryProdInfoAdo**, а затем выберите команду **Copy**.
8. В окне CPP-конвейера щелкните правой кнопкой мыши на элементе **Requisition Item Info** (Информация о единице номенклатуры заявки), а затем выберите команду **Paste Component** (Вставить компонент). При этом компонент `QueryProdInfoAdo` будет помещен в ступень Requisition Item Info.
9. Повторите шаги 7 и 8 для компонентов, перечисленных в табл. 3.14.

Таблица 3.14. *Данные настройки компонентов*

Компонент P-конвейера	Компонент CPP-конвейера	Ступень конвейера
Проверка заказа (`Order Check`)	`Validate Ship-To`	`Requisition Check`
Согласованная стоимость единицы товара (`Item Adjust Price`)	`SaleAdjust`	Согласованная стоимость единицы номенклатуры заявки (`Requisition Item Adjust Price`)
Согласованная стоимость единицы номенклатуры заявки (`Requisition Item Adjust Price`)	`DbOrderPromoAdo`	Согласованная стоимость заявки (`Requisition Adjust Price`)

10. Щелкните по компонентам правой кнопкой мыши, а затем выберите команду **Delete** для удаления существующих компонентов в ступени Shipping (Отгрузка) в CPP-конвейере.
11. Скопируйте три компонента `FixedShipping` ступени Shipping в P-конвейере в ступень Shipping CPP-конвейера.

Примечание

Порядок компонентов в ступени Shipping CPP-конвейера должен быть такой же, как иерархия ступени Shipping P-конвейера.

12. В меню **File** выберите команду **Save As** для сохранения CPP-конвейера под папкой C:\InetPub\wwwroot\FiveLakes\Config. Сохраните данный файл под именем CorpPlan.pcf. Когда появится соответствующее приглашение, выберите опцию для записи поверх существующего файла.

Примечание

Убедитесь, что имя конвейера такое же, как в тексте. Не используйте имя, присвоенное по умолчанию.

Для ознакомления с файлами i_util.asp и xt_orderform_purchase.asp сайта FiveLakes:

1. Откройте ASP-страницу i_util.asp в Microsoft Script Debugger. Этот файл находится в папке C:\InetPub\wwwroot\FiveLakes.
2. В функции `UtilRunPlan` найдите строку кода:

```
errorLevel = UtilRunPipe("CorpPlan.pcf", mscsOrderForm, _
  mscsPipeContext)
```

3. Откройте страницу xt_orderform_purchase.asp в Microsoft Script Debugger. Этот файл размещается в корневой папке сайта FiveLakes.
4. В функции `OrderFormPurchase` найдите строку кода:

```
errorLevel = UtilRunPipe("CorpPlan.pcf", mscsOrderForm, _
  mscsPipeContext)
```

Упражнение 2. Создание CPS-конвейера

В этом упражнении вы создадите CPS-конвейер и напишете код для его активизации на созданном для примера Web-сайте FiveLikes.

Для того чтобы создать CPS-конвейер на сайте FiveLakes:

1. В редакторе конвейера (Pipeline Editor) в меню **File** выберите команду **New**. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) отобразится список имеющихся шаблонов.
2. Выберите файл CorpPurchaseSubmit.pct и затем нажмите кнопку **OK**.

Для того чтобы вставить компоненты `MakePO` и `POtoFile` в ступень Purchase Order Submit (Отправить заказ на закупку):

1. Щелкните правой кнопкой мыши на элементе **Purchase Order Submit**, а затем выберите команду **Insert Component** (Вставить компонент). В диалоговом окне **Choose a component** (Выберите компонент) отображаются компоненты, которые могут быть включены в этот этап.
2. Выберите элемент **Make PO**, а затем нажмите кнопку **OK** для включения этого компонента в данную ступень.
3. Щелкните правой кнопкой мыши на элементе **Make PO**, а затем выберите команду **Properties**. Появится диалоговое окно **Properties**.
4. Для создания заказа на закупку (PO, Purchase Order) с помощью компонента `MakePO` будет использоваться файл шаблона. Нажмите кнопку **Browse**, чтобы выбрать файл шаблона, укажите файл POTemplate.txt в папке Config сайта FiveLakes, а затем нажмите кнопку **Open**.
5. Нажмите кнопку **OK** для подтверждения установок.
6. Далее вставьте компонент `POtoFile` в ступень Purchase Order Submit. Щелкните правой кнопкой мыши на элементе **Make PO**, выберите команду **Insert Component** (Вставить компонент), а затем щелкните **After** (После). В диалоговом окне **Choose a component** (Выберите компонент) будет отображен список компонентов, которые можно вставить.
7. Выберите элемент **POtoFile**, а затем нажмите кнопку **OK**, чтобы включить данный компонент в этап.
8. Щелкните правой кнопкой мыши на элементе **POtoFile**, а затем выберите команду **Properties**.

9. Нажмите кнопку **Browse** для указания имени файла, в который будет записан заказ на закупку. Измените **Destination File Name** (Имя файла по умолчанию) на папку FiveLakes, задайте имя файла — PO_Data.txt, а затем нажмите кнопку **Save**.
10. Щелкните **Append to file instead of overwriting file** (Приписать к файлу, а не переписать файл заново) и затем нажмите кнопку **OK** для подтверждения установок.

Для включения компонента `Scriptor` в ступень отправки заказа на закупку (Purchase Order Submit):

1. Щелкните правой кнопкой мыши на элементе **POtoFile**, выберите команду **Insert Component** (Вставить компонент), а затем щелкните **After** (После).
2. В диалоговом окне **Choose a component** (Выберите компонент) выберите **Scriptor**, а затем нажмите кнопку **OK** для включения данного компонента в ступень.
3. Щелкните правой кнопкой на элементе **Scriptor**, а затем выберите команду **Properties**.
4. В диалоговом окне **Component Properties** (Свойства компонента) на вкладке **Scriptor** щелкните **External** (Внешний).
5. В окне подтверждения **Scriptor** нажмите кнопку **No**. Вы будете использовать внешний скрипт-файл.
6. Нажмите кнопку **Browse** для установки имени и пути файла. Файл CallTransmit.vbs находится в папке Config сайта FiveLakes. Этот файл будет вызывать конвейер для передачи (Transmit Pipeline). В диалоговом окне **Open** выберите CallTransmit.vbs, а затем нажмите кнопку **Open**.
7. Нажмите кнопку **OK** для подтверждения произведенных установок.
8. В меню **File** укажите команду **Save As** для сохранения CPS-конвейера. Измените указанный путь на C:\InetPub\wwwroot\FiveLakes\Config. В диалоговом окне **Save As** введите CorpSubmit.pcf, а затем нажмите кнопку **Save**. Когда появится приглашение, выберите **write over the existing file** (запись поверх существующего файла).

Примечание

Убедитесь в том, что конвейеру присвоено точно такое же имя, как указано в данном тексте.

Для того чтобы ознакомиться со страницами xt_orderform_purchase.asp сайта FiveLakes, в странице xt_orderform_purchase.asp найдите следующий текст:

```
errorLevel = UtilRunTxPipe("purchaseupdate.pcf", mscsOrderForm, _
  mscsPipeContext)
else
  REM Run the transacted pipe
```

```
  errorLevel = UtilRunTxPipe("purchaseinsert.pcf", mscsOrderForm, _
  mscsPipeContext)
end if
errorLevel = UtilRunSubmit("CorpSubmit.pcf", mscsOrderForm, _
mscsPipeContext)
```

Этот код активизирует конвейеры для закупок (purchasing) и передачи (submit).

Упражнение 3. Создание CIT-конвейера

В этом упражнении вы сформируете CIT-конвейер, служащий для отправки заказа, на созданном для примера Web-сайте FiveLakes. Для передачи заказа на закупку на сайт Ramona Publishing используется компонент `SendHTTP`.

Для создания CIT-конвейера на сайте FiveLakes:

1. В редакторе конвейера (Pipeline Editor) в меню **File** выберите команду **New**. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) отображается список доступных шаблонов.
2. Выберите файл transmit.pct и затем нажмите кнопку **OK**.

Для включения компонента `MapToXML` в ступень конвертирования (Map stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Map** (Конвертировать), а затем выберите команду **Insert Component** (Вставить компонент). В диалоговом окне **Choose a component** (Выберите компонент) отображается список имеющихся компонентов.
2. Выберите элемент **MapToXML**, а затем нажмите кнопку **OK** для вставки компонента `Map` в конвейер.
3. Щелкните правой кнопкой мыши на компоненте **MapToXML**, а затем выберите команду **Properties**. Необходимо использовать установки по умолчанию. Ключом источника объекта (object source key) по умолчанию является **object**, ключом результата (results xml key) — **working_data**, а формат данных — XML.
4. Нажмите кнопку **OK**, чтобы закрыть диалоговое окно **Properties**.

Для того чтобы включить компонент `Add an AddHeader` в ступень включения заголовка (Add Header stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Add Header** (Добавить заголовок), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) выберите компонент `AddHeader` и затем нажмите кнопку **OK** для того, чтобы включить этот компонент в данную ступень.

3. Щелкните **AddHeader**, а затем укажите **Properties**.
4. Нужно использовать установки, заданные по умолчанию. В поле ввода (input) и вывода (output) установлено значение **working_data**.
5. Нажмите кнопку **OK**, чтобы закрыть диалоговое окно **Properties**.

Для включения компонента `Add a SendHTTP` в транспортную ступень (Transport stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Transport**, а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** выберите **SendHTTP**, а затем нажмите кнопку **OK** для добавления выбранного компонента в данную ступень.
3. Щелкните правой кнопкой мыши на элементе **SendHTTP**, а затем выберите команду **Properties**.
4. В поле URL введите: **localhost/Ramona/getdata.asp**. Указанный URL вызовет страницу getdata.asp сайта Ramona.
5. Введите `working_data` в окне **Field to be posted** (Поле отправки), а затем нажмите кнопку **OK** для подтверждения данных установок.
6. В меню **File** выберите команду **Save As** для того, чтобы сохранить CIT-конвейер. Сохраните данный файл под именем transmit.pcf в папке C:\InetPub\wwwroot\FiveLakes\Config. Когда появится соответствующее приглашение, выберите опцию для записи поверх существующего файла (**write over the existing file**).

Примечание

Убедитесь в том, что конвейеру присвоено точно такое имя, как указано в данном тексте.

Вопросы для повторения

1. Какие функции выполняет CPP-конвейер (Corporate Purchasing Plan pipeline)?
2. Какие функции выполняет CPS-конвейер (Corporate Purchasing Submit pipeline)?
3. Под каким именем объявляется бизнес-объект в шаблоне `MakePO`?
4. Какие компоненты могут использоваться для передачи данных в транспортной ступени CIT-конвейера?
5. Какие CI-конвейеры применяются для взаимодействия класса B2B?
6. В каких ступенях CIT-конвейера вы можете применять сертификаты?

ГЛАВА 4

Получение заказа

Краткий обзор

Конвейер приема или *CIR-конвейер* (Receive pipeline) является формой CI-конвейера (Commerce Interchange Pipeline). CIR-конвейер осуществляет прием информации, переданной электронными партнерами. В этой главе вы ознакомитесь со ступенями данного конвейера. Вы также узнаете, как принять заказ на закупку и *передать квитанцию подтверждения*[1] (receipt).

Цели и задачи

После прочтения данной главы вы научитесь:

- описывать ступени в CIR-конвейере;
- создавать CIR-конвейер для приема заказа на закупку;
- передавать квитанцию подтверждения отправителю;
- принимать квитанцию подтверждения на сайте отправителя.

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod04\Mod04Code.txt.

Получение заказа на закупку

В типовой бизнес-транзакции компания посылает заказ на закупку поставщику, который затем обрабатывает этот заказ. В транзакциях электронной

[1] Имеется в виду квитанция подтверждения приема информации. — *Пер.*

коммерции поставщики принимают заказы при помощи так называемого конвейера приема (CIR-конвейера), который является формой CI-конвейера.

Далее вы познакомитесь со ступенями этого конвейера. Вы изучите, как осуществляется прием заказов на закупку при помощи CIR-конвейера, а также передача и прием квитанции подтверждения.

Ступени в CIR-конвейере

Бизнес в электронной коммерции класса B2B представляет собой двусторонний процесс. Сначала электронный партнер инициирует транзакцию, посылая поставщику заказ на закупку в виде объекта данных, используя для этого CIT-конвейер.

На рис. 4.1 показаны семь ступеней, имеющихся в CIR-конвейере.

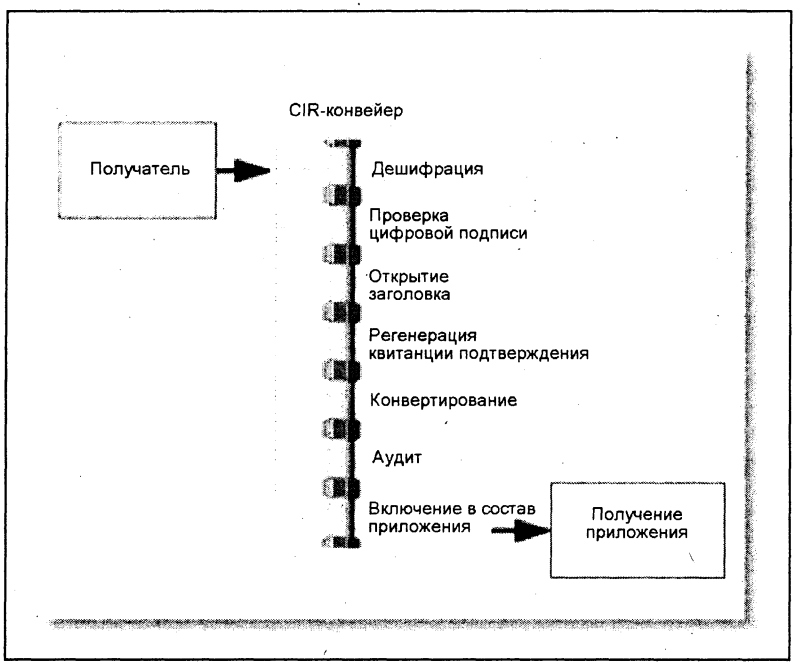


Рис. 4.1. Ступени CIR-конвейера

CIR-конвейер распаковывает данные, хранящиеся в транспортном словаре, и включает эти данные в назначенное приложение. Данный конвейер состоит из следующих ступеней:

- дешифрация (Decrypt);
- проверка цифровой подписи (Verify Digital Signature);

- открытие заголовка (Open Header);
- генерация квитанции подтверждения (Generate Receipt);
- конвертирование (Map);
- аудит (Audit);
- включение в состав приложения (Application Integration).

Дешифрация

Для обеспечения безопасности информации бизнес-данные прежде, чем будут переданы, обычно шифруются. Зашифрованные данные должны расшифровываться на стороне получателя перед тем, как к ним будет предоставлен доступ. Ступень дешифрации обеспечивает расшифровку данных и затем обновляет рабочие данные (working data) в транспортном словаре, помещая туда эти расшифрованные данные. Для дешифрации данных используется один из компонентов, представленных в табл. 4.1.

Таблица 4.1. *Описание компонентов, используемых для дешифрации*

Компонент	Описание
`DecodeSMIME`	Компонент декодирует поток данных, который был инкапсулирован в формат S/MIME (secure MIME), представленный в draft-стандарте Internet IETF EDIINT, и записывает результат в область рабочих данных (working data)
`DecryptPKCS`	Компонент дешифрирует поток данных, используя стандарт PKCS7, и записывает результат в область рабочих данных (working data)

Проверка цифровой подписи

Ступень проверки цифровой подписи подтверждает цифровую подпись электронного партнера. В эту ступень входят два компонента (табл. 4.2).

Таблица 4.2. *Компоненты для проверки цифровой подписи*

Компонент	Описание
`DecodeSMIME`	Компонент декодирует объект данных, который был инкапсулирован в формат S/MIME (secure MIME) draft-стандарта Internet IETF EDIINT
`VerifyDigitalSig`	Компонент верифицирует цифровую подпись данных, хранимых в транспортном словаре, используя стандарт PKCS7

Открытие заголовка

Когда данные переданы, ступень присоединения заголовка (Add Header) CIT-конвейера упаковывает рабочие данные. Ступень открытия заголовка (Open Header) CIR-конвейера выполняет обратные действия в сравнении с теми, которые были осуществлены компонентами ступени включения заголовка CIT-конвейера. Атрибуты сообщения данных хранятся в транспортном словаре, пока бизнес-данные открыты для ступени конвертирования (Map) CIR-конвейера.

Компонент `OpenHeader` создает *комплексное сообщение* (digest) в том случае, если отправителем был получен запрос на квитанцию подтверждения. Комбинированное сообщение подтверждает, что объект данных был получен, и оно передается отправителю с помощью ступени генерации квитанции подтверждения (Generate Receipt stage). Эта ступень содержит компоненты, перечисленные в табл. 4.3.

Таблица 4.3. *Компоненты ступени открытия заголовка*

Компонент	Описание
`DecodeMIME`	Компонент декодирует объект данных, который был упакован (encapsulated) в MIME-формат
`OpenHeader`	Компонент делает синтаксический анализ XML-данных, присоединенных к объекту данных, с помощью ступени присоединения заголовка (Add Header) CIT-конвейера. Он записывает идентификатор транзакции (transaction ID), дату и время передачи, а также, в необязательном порядке, комбинированное сообщение в транспортный словарь в виде пар "имя/значение". Могут быть записаны в качестве опции адреса отправителя и получателя

Генерация квитанции подтверждения

Если отправитель запросил так называемую *квитанцию подтверждения* (receipt), ступень генерации квитанции подтверждения (Generate Receipt) создает объект данных квитанции подтверждения, а затем инициирует CIT-конвейер для передачи этой квитанции. Данная ступень содержит один компонент (табл. 4.4).

Таблица 4.4. *Компонент ступени генерации квитанции подтверждения*

Компонент	Описание
`GenerateReceipt`	Использует комбинированное сообщение, созданное компонентом `OpenHeader`. Компонент `GenerateReceipt` считывает значения, записанные в транспортный словарь. Если запрашивалась квитанция подтверждения, то данный компонент создает эту квитанцию и инициирует CIT-конвейер для отправки квитанции подтверждения в соответствии с запросом

Примечание

Исходный отправитель обеспечивает CIR-конвейер для приема квитанции подтверждения. Указанный конвейер может содержать компонент `AuditReceipt`, который записывает эту квитанцию в базу данных аудита.

Конвертирование

Ступень конвертирования транслирует рабочие данные, хранящиеся в транспортном словаре, обратно — в исходный объект данных. Эта ступень содержит один компонент (табл. 4.5).

Таблица 4.5. *Компонент ступени конвертирования*

Компонент	Описание
`MapFromXML`	Транслирует данные из потока XML-данных обратно в исходный объект данных, а после трансляции данных записывает объект данных в транспортный словарь

Аудит

Ступень аудита сохраняет информацию о бизнес-транзакции в базе данных аудита и содержит компоненты, перечисленные в табл. 4.6.

Таблица 4.6. *Компоненты ступени аудита*

Компонент	Описание
`Audit`	Компонент считывает определенные значения из транспортного словаря и записывает их в таблицу базы данных аудита
`AuditReceipt`	Компонент записывает данные из квитанции в таблицу базы данных аудита

Включение в состав приложения

Ступень включения в состав приложения (Application Integration stage) содержит компоненты, осуществляющие конвертирование объектов данных в формат, который может использоваться в конечном приложении (target application). Эта ступень содержит один компонент (табл. 4.7).

Таблица 4.7. *Компонент ступени включения в состав приложения*

Компонент	Описание
`SQLOrderADO`	Выполняет SQL-запрос

Этапы приема заказа

Когда электронный партнер посылает заказ на закупку поставщику, тот должен создать конвейер приема (CIR-конвейер) прежде, чем начнет принимать этот заказ. Например, если Web-сайт FiveLakes передает заказ в компанию "Ramona Publishing", то сайт Ramona должен создать CIR-конвейер, чтобы получить возможность принять заказ, выбрать необходимые данные из него и создать квитанцию подтверждения.

На рис. 4.2 представлены ступени приема заказа на закупку.

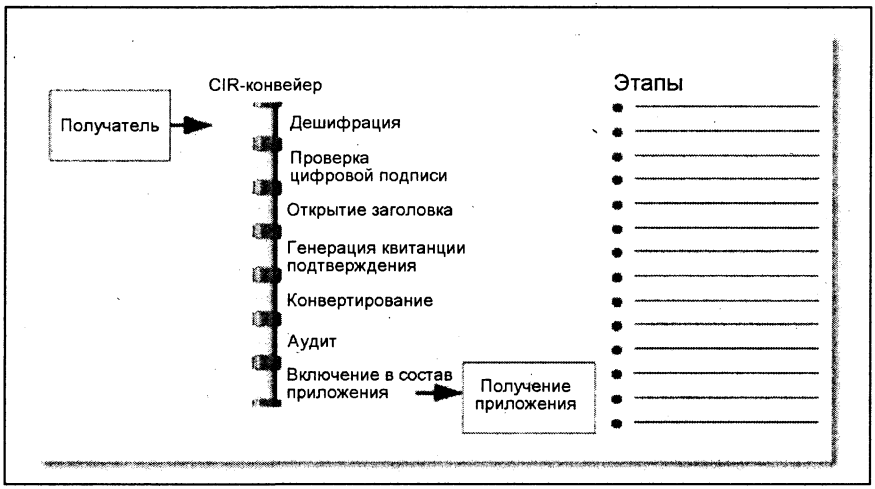


Рис. 4.2. Ступени приема заказа на закупку

Компонент `SendHTTP` использовался для передачи заказа на закупку на сайт компании "Ramona". Этот компонент вызывает ASP-страницу getdata.asp, которую выполняет система заказчика, для приема подтверждения, отправляемого в соответствии с протоколом HTTP.

На сайте отправителя (FiveLakes) был создан транспортный словарь для передачи формы заказа в CIT-конвейер. Этот транспортный словарь содержит форму заказа. Компоненты CIT-конвейера обновляют транспортный словарь, записывая в него промежуточные значения.

Вызов транспортного словаря

Сайт получателя (Ramona) создаст транспортный словарь, который будет передан в качестве аргумента в CIR-конвейер. Для вызова транспортного словаря вам необходимо выполнить следующие шаги:

- создать транспортный словарь;
- произвести выборку информации с помощью метода `Request.Form`;

- создать Т-конвейер (Transacted pipeline);
- создать контекстный словарь (Context dictionary);
- выполнить данный конвейер.

Создание транспортного словаря

ASP-страница getdata.asp на сайте Ramona содержит код, необходимый для создания транспортного словаря. Вызов страницы getdata.asp осуществляет компонент SendHTTP в Т-конвейере сайта компании "FiveLakes". Код ASP-страницы getdata.asp имеет следующий вид:

```
Dim TransportDictionary
Set TransportDictionary = Server.CreateObject("Commerce.Dictionary")
```

Выборка информации с помощью метода Request.Form

Для осуществления выборки информации, отправленной на сайт, используется метод `Request.Form`. В транспортном словаре эта информация хранится в виде пар "имя/значение".

```
TransportDictionary.working_data = Request.Form
```

Создание Т-конвейера

Теперь создается Т-конвейер (`mscsMtsTxPipeline`) для загрузки CIR-конвейера (Receive pipeline).

```
Set pipe = Server.CreateObject("Commerce.MtsTxPipeline")
Call pipe.loadpipe("receive.pcf")
```

Создание контекстного словаря и обеспечение выполнения конвейера

Далее создается контекстный словарь (Context dictionary) и выполняется конвейер.

```
Set context = Server.CreateObject("Commerce.Dictionary")
errorlevel = pipe.Execute(1, TransportDictionary,context, 0)
```

При получении объекта CIR-конвейер распаковывает его, используя для этого следующие ступени: дешифрации (Decrypt), проверки цифровой подписи (Verify Digital Signature), открытия заголовка (Open Header), генерации квитанции подтверждения (Generate Receipt), конвертирования (Map), аудита (Audit) и включения в состав приложения (Application Integration).

Обратите внимание, что CIR-конвейер выполняет те же действия, что и CIT-конвейер, только в обратном порядке.

Ступень дешифрации

Ступень дешифрации осуществляет дешифрирование объекта данных, если этот объект был зашифрован CIT-конвейером на стороне электронного партнера.

Ступень проверки цифровой подписи

Ступень проверки цифровой подписи подтверждает цифровую подпись электронного партнера. Например, компонент `VerifyDigitalSig` данной ступени для проверки цифровой подписи объекта данных использует сертификат отправителя объекта данных.

Примечание

Конструкция CIT-конвейера должна быть зеркально отображена на принимающей стороне. Например, если компонент `EncodeSMIME` входил в состав CIT-конвейера, CIR-конвейер должен содержать компонент `DecodeSMIME`. Если CIT-конвейер содержит компонент `DigitalSig`, то CIR-конвейер должен включать компонент `VerifyDigitalSig`.

Ступень открытия заголовка (Open Header) содержит компонент `OpenHeader`. Этот компонент считывает XML-теги, приписанные к рабочим данным при помощи компонента вставки заголовка (AddHeader) в CIT-конвейере, а затем записывает их в виде пар "имя/значение" в транспортный словарь. Если включен запрос на квитанцию подтверждения, то компонент `OpenHeader` хэширует (hashes) объект данных для того, чтобы создать так называемое *комбинированное сообщение* (digest). Комбинированное сообщение не предназначено для прочтения, а служит в качестве квитанции подтверждения. Комбинированное сообщение также записано в виде пар "имя/значение" в транспортный словарь.

Компонент `OpenHeader` осуществляет подготовку транспортного словаря для следующей ступени — генерации квитанции подтверждения (Generate Receipt).

Ступень генерации подтверждения

В ступени генерации подтверждения используется компонент `GenerateReceipt`, предназначенный для чтения значений, записанных компонентом `OpenHeader`. Этот компонент создает объект квитанции подтверждения и затем посылает его к исходному отправителю с помощью нового CIT-конвейера. Компонент `GenerateReceipt` не посылает все значения, хранящиеся в исходном транспортном словаре, — он отправляет лишь *идентификатор транзакции* (transaction ID), дату и время, относящиеся к принимающей системе, комбинированное сообщение (digest), созданное компонентом `OpenHeader`, и, в необязательном порядке, тип документа, источник и назначение, которые описаны в исходной передаче.

Путь pcf-файла отправки и приема квитанции подтверждения компонента `GenerateReceipt` указывает полный путь файла конфигурации CIT-конвейера. Этот конвейер используется для отправки квитанции подтверждения исходному отправителю. Затем исходный отправитель предоставляет CIR-конвейер для приема подтверждения. Этот конвейер содержит компонент, который записывает квитанцию подтверждения в таблицу базы данных аудита.

Ступень конвертирования

В ступени подтверждения (Map) компонент `MapFromXML` конвертирует поток XML-данных обратно в объект данных. Возвращенный таким образом объект данных относится к тому же типу данных, к которому он принадлежал до передачи. Приложение, осуществляющее прием, использует этот итоговый объект данных.

Компонент `MapFromXML` может конвертировать данные из любого объекта данных, в котором реализуется интерфейс `IPersistStreamInit` или же интерфейс `IPersistXML`. Это следующие объекты: Commerce Server Dictionary, SimpleList, OrderForm, а также Active Data Objects (ADO) Recordset.

Ступень включения в состав приложения

Ступень включения в состав приложения (Application Integration stage) служит для содержания компонентов, конвертирующих объекты данных в формат, который, в свою очередь, может быть прочитан оконечным приложением (target application). Например, вы можете включить компонент `Scriptor` в эту ступень для вызова внешнего скрипта.

Данный скрипт осуществляет связь с базой данных поставщика и записывает значения из транспортного словаря в таблицу, как представлено в следующем фрагменте кода:

```
Function mscsexecute(config, transportdictionary, context, flags)
  Dim Connection
  Set Connection = CreateObject("ADODB.Connection")
  connection.provider="sqloledb"
  Connection.Open "datasource=ramona;initial "& _
                  "catalog=ramona;user id=sa; password=;"
  Dim success
  success = VendorInsertitems(transportdictionary.working_data, _
            Connection,Context)
  If success Then
    mscsexecute = 1
  Else
    mscsexecute = 3
  End If
```

```
  Connection.close
End Function

Function VendorInsertitems(ByRef working_data, ByRef _
  Connection,ByRef Context)
  VendorInsertitems=True
  dim rs,po_no
  set rs=CreateObject("ADODB.Recordset")
  rs.Open "ramona_po",connection,2,2
  If rs.EOF Then
    po_no=1
  Else
    rs.movelast
    po_no=rs("po_no")
    po_no=po_no + 1
  End If
  For Each item In working_data.items
    rs.addnew
    rs("po_no")=po_no
    rs("sku")=item.sku
    rs("name")=item.name
    rs("quantity")=item.quantity
    rs.update
  Next
  rs.close
  If (0 < Connection.Errors.Count) Then
    VendorInsertitems = False
  End If
End Function
```

Этот код осуществляет ADODB-связь с базой данных компании "Ramona" и открывает таблицу `Ramona_PO` в этой базе данных. Скрипт генерирует номер заказа на закупку и сохраняет информацию относительно данного заказа в указанной таблице.

Лабораторная работа 4. Прием заказа на закупку

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать CIR-конвейер для приема заказа на закупку;
- принимать заказ на закупку с помощью CIR-конвейера;

- передавать квитанцию подтверждения исходному отправителю;
- осуществлять прием квитанции на сайте исходного отправителя.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь основные понятия об HTML и ASP-скриптах;
- быть знакомым с Microsoft Internet Explorer и Microsoft Visual InterDev 6.0;
- выполнить упражнения, представленные в лабораторной работе 3.

Подготовка к лабораторной работе

Для выполнения данной работы вы должны скопировать файлы из папки <папка_установки>\Labs\Lab04\Ramona\StartCode в папку C:\Inetpub\wwwroot\Ramona.

Примечание

Время, необходимое для выполнения данной работы, — 45 минут.

Упражнение 1. Прием заказа с помощью CIR-конвейера

В этом упражнении вы осуществите прием заказа на закупку на Web-сайте Ramona Publishing.

Для того чтобы открыть файл getdata.asp в каталоге сайта, воспользуйтесь программой Microsoft Script Debugger и откройте файл getdata.asp, который расположен по адресу C:\Inetpub\wwwroot\Ramona.

Для того чтобы ознакомиться со скриптом, который вызывает CIR-конвейер:

1. Найдите следующий фрагмент кода в файле getdata.asp. Он создает транспортный словарь и вызывает CIR-конвейер:

```
<%
  Dim TransportDictionary
  set TransportDictionary = _
  server.createobject("commerce.dictionary")
  TransportDictionary.working_data = request.form
  ' Create a Pipeline Object, load a PCF
  Set pipe = Server.CreateObject("Commerce.MtsPipeline")
  call pipe.loadpipe("c:\inetpub\wwwroot\ramona\config\"& _
                     "receive.pcf")
```

```
    'Create a context dictionary and execute
    Set context = Server.CreateObject("Commerce.Dictionary")
    errorlevel = pipe.Execute(1, TransportDictionary, context, 0)
%>
```

2. Закройте файл.

Для запуска редактора конвейера (Pipeline Editor) нажмите кнопку **Start**, выберите пункт **Programs** и, далее, **Microsoft Site Server** и **Commerce**, а затем укажите команду **Pipeline Editor**.

Для создания CIR-конвейера на сайте компании "Ramona":

1. В меню **File** выберите команду **New**.
2. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) укажите файл receive.pct, а затем нажмите кнопку **OK**.

Для включения компонента `OpenHeader` в ступень Open Header CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Open Header** (Открыть заголовок), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) выберите **OpenHeader**, а затем нажмите кнопку **OK**.

Для данного компонента будут использоваться установки, принятые по умолчанию.

Для того чтобы включить компонент `GenerateReceipt` в ступень генерации квитанции подтверждения (Generate Receipt stage) CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Generate Receipt** (Создать квитанцию подтверждения), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) укажите **GenerateReceipt**, а затем нажмите кнопку **OK**.
3. Щелкните правой кнопкой мыши на элементе **GenerateReceipt**, а затем выберите команду **Properties**.
4. На вкладке **GenerateReceipt** введите `C:\InetPub\wwwroot\Ramona\Config\transmit.pcf` в поле **Send Receipt PCF Filepath** (Путь pcf-файла отправки квитанции подтверждения), а затем нажмите кнопку **OK**.

Для того чтобы вставить компонент `MapFromXML` в ступень конвертирования (Map) CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Map** (Конвертирование), а затем выберите команду **Insert Component** (Вставить компонент).

2. В диалоговом окне **Choose a component** (Выберите компонент) укажите **MapFromXML**, а затем нажмите кнопку **OK**.
3. Щелкните правой кнопкой мыши на элементе **MapFromXML**, а затем выберите команду **Properties**.
4. На вкладке **Map From XML** измените значение **Result Object Key** (Ключ итогового объекта) с **object** на **working_data**, а затем нажмите кнопку **OK**.

Для того чтобы вставить компонент `Scriptor` в ступень включения в состав приложения (Application Integration stage) CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Application Integration**, а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) выберите **Scriptor**, а затем нажмите кнопку **OK**.
3. Щелкните правой кнопкой мыши на элементе **Scriptor**, а затем выберите команду **Properties**.
4. На вкладке **Scriptor** щелкните **External**.
5. В окне подтверждения **Scriptor** нажмите кнопку **No**. Тем самым вы укажите, что не хотите экспортировать внутренний исходный код.
6. Нажмите кнопку **Browse** для того, чтобы выбрать соответствующий файл скрипта.
7. Измените значение пути так, чтобы была указана папка Config на Web-сайте компании "Ramona".
8. Выберите файл POAccept.vbs, а затем нажмите кнопку **Open**.
9. Нажмите кнопку **OK** для подтверждения произведенных установок.
10. Сохраните конвейер в виде файла receive.pcf в папке C:\Inetpub \wwwroot\Ramona\Config. Когда появится соответствующее приглашение, запишите файл поверх уже существующего файла.

Упражнение 2. Передача квитанции подтверждения на сайт FiveLakes

В этом упражнении вы создадите CIT-конвейер, предназначенный для передачи квитанции подтверждения на сайт компании "FiveLakes".

Для того чтобы создать CIT-конвейер на сайте компании "Ramona":

1. Запустите программу Pipeline Editor.
2. В меню **File** выберите команду **New**. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) отображается список имеющихся шаблонов.
3. Выберите файл transmit.pct, а затем нажмите кнопку **OK**.

Для включения компонента `MapToXML` в ступень конвертирования (Map stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Map** (Конвертирование), а затем укажите команду **Insert Component** (Вставить компонент). В диалоговом окне **Choose a component** (Выберите компонент) отображается список компонентов.
2. Выберите **MapToXML**, а затем нажмите кнопку **OK** для включения данного компонента в ступень.
3. Щелкните правой кнопкой мыши на элементе **MapToXML**, а затем выберите команду **Properties**. Необходимо использовать установки, принятые по умолчанию. Таким значением для **Object Source Key** является **object**, а для **Results XML Key** — **working_data**, а в качестве формата данных (preferred data format) — теги XML.
4. Нажмите кнопку **OK**, чтобы закрыть диалоговое окно.

Для того чтобы включить компонент `Add the AddHeader` в ступень включения заголовка (Add Header stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Add Header** (Включить заголовок), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) укажите компонент **AddHeader**, а затем нажмите кнопку **OK** для включения его в ступень.
3. Щелкните правой кнопкой мыши на элементе **AddHeader**, а затем выберите команду **Properties**. Должны использоваться установки, принятые по умолчанию. В качестве области для ввода и вывода используется **working_data**.
4. Нажмите кнопку **OK** для закрытия диалогового окна.

Для включения компонента `SendHTTP` в транспортную ступень (Transport stage) CIT-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Transport**, а затем выберите команду **Insert Component**.
2. В диалоговом окне **Choose a component** (Выберите компонент) выберите компонент **SendHTTP**, а затем нажмите кнопку **OK** для включения его в данную ступень.
3. Щелкните правой кнопкой мыши на элементе **SendHTTP**, а затем укажите команду **Properties**.
4. В поле URL введите **localhost/FiveLakes/accreceipt.asp**. Указанный URL вызовет файл accreceipt.asp на сайте FiveLakes.

5. В окне **Field to be posted** введите `working_data`, а затем нажмите кнопку **OK** для подтверждения данных установок.
6. В меню **File** выберите команду **Save** для сохранения CIT-конвейера. Сохраните данный файл как transmit.pcf в папке C:\InetPub\wwwroot\Ramona\Config. Когда появится соответствующее приглашение, запишите файл поверх уже существующего файла.

Упражнение 3. Прием квитанции подтверждения с сайта Ramona

В этом упражнении вы создадите CIR-конвейер на сайте FiveLakes для приема квитанции подтверждения от сайта компании "Ramona".

Для того чтобы ознакомиться со скриптом, предназначенным для запуска CIR-конвейера на сайте FiveLakes:

1. Откройте файл accreceipt.asp, расположенный в папке C:\InetPub\wwwroot\FiveLakes, с помощью программы Microsoft Script Debugger.
2. Просмотрите следующий код, который создает транспортный словарь и запускает CIR-конвейер:

```
<%
  Dim TransportDictionary
  set TransportDictionary = _
  server.createobject("commerce.dictionary")
  TransportDictionary.working_data = request.form
  ' Для создания Pipeline Object, загружает PCF-файл
  Set pipe = Server.CreateObject("Commerce.MtsPipeline")
  call pipe.loadpipe("c:\inetpub\wwwroot\fivelakes\config\"& _
                     "receive.pcf")
  ' Создает контекстный словарь и осуществляет выполнение
  Set context = Server.CreateObject("Commerce.Dictionary")
  errorlevel = pipe.Execute(1, TransportDictionary, context, 0)
%>
```

3. Закройте файл.

Для запуска редактора конвейера Pipeline Editor нажмите кнопку **Start**, выберите пункт **Programs** и, далее, **Microsoft Site Server** и **Commerce**, а затем укажите команду **Pipeline Editor**.

Для создания CIR-конвейера:

1. В меню **File** выберите команду **New**.
2. В диалоговом окне **Choose a Pipeline Template** (Выберите шаблон конвейера) выберите receive.pct, а затем нажмите кнопку **OK**.

Для включения компонента `OpenHeader` в ступень открытия заголовка (Open Header stage) CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Open Header** (Открыть заголовок), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) укажите **OpenHeader**, а затем нажмите кнопку **OK**. Для данного компонента будут использоваться установки, принятые по умолчанию.

Для включения компонента `MapFromXML` в ступень Map CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Map** (Конвертирование), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) укажите **MapFromXML**, а затем нажмите кнопку **OK**. Для данного компонента будут использоваться установки, принятые по умолчанию.

Для вставки компонента `AuditReceipt` в ступень аудита (Audit stage) CIR-конвейера:

1. Щелкните правой кнопкой мыши на элементе **Audit** (Аудит), а затем выберите команду **Insert Component** (Вставить компонент).
2. В диалоговом окне **Choose a component** (Выберите компонент) укажите **AuditReceipt**, а затем нажмите кнопку **OK**.
3. Щелкните правой кнопкой мыши на элементе **AuditReceipt**, а затем выберите команду **Properties**.
4. На вкладке **AuditReceipt**:
   - введите `DSN=FiveLakes;UID=sa;PWD=;` в поле **Connection String** для указания связи;
   - введите `FiveLakes_POReceipt` в поле **Table** для указания таблицы, предназначенной для хранения квитанции подтверждения. Таблица `FiveLakes_POReceipt` уже была создана в базе данных `FiveLakes`.
5. Нажмите кнопку **OK** для подтверждения данных установок.
6. Сохраните данный конвейер как файл receive.pcf в папке C:\InetPub\wwwroot\FiveLakes\Config. После соответствующего приглашения запишите данный файл вместо существующего файла.

Упражнение 4. Покупки на сайте FiveLakes

Допустим, вы являетесь заказчиком и помещаете свой заказ на Web-сайте компании "FiveLakes". Ввиду того, что компания "FiveLakes" не осуществляет управления запасами (товарами), все заказы на закупку автоматически поступают на сайт электронной коммерции компании "Ramona Publishing".

При получении заказа на закупку с сайта компании "FiveLakes" сайт компании "Ramona" обрабатывает его и генерирует квитанцию подтверждения, которую передает на сайт FiveLakes. Заказ на закупку сохраняется в таблице `Ramona_PO` базы данных `Ramona`. Квитанция, отправленная сайтом компании "Ramona", сохраняется в таблице `FiveLakes_POReceipt` базы данных `FiveLakes`.

В этом упражнении вы сымитируете делать покупку на сайте компании "FiveLakes" в качестве демонстрации применения CIT- и CIR-конвейеров.

Для осуществления покупки на сайте FiveLakes:

1. Откройте Microsoft Internet Explorer, а затем введите следующий URL в окне **Address**: **http://localhost/fivelakes/**.
2. Щелкните по гиперссылке **Computers**.
3. Выберите категорию `Computer Networks` для просмотра подробной информации о приобретаемом изделии.
4. Щелкните по кнопке-изображению **Buy Now** (Купить сейчас), чтобы включить данное изделие в покупательскую корзину.
5. Щелкните **Continue to Checkout** (Продолжить для подсчета стоимости покупки), чтобы приобрести данное изделие.
6. Нажмите кнопку **Total** (Итоговая сумма) на странице **Shipping** (Отгрузка).
7. Нажмите кнопку **Purchase** (Закупка) на странице **Payment** (Платеж).

Для просмотра заказа на закупку, полученного на сайте компании "Ramona", а так же квитанции подтверждения, полученной на сайте компании "FiveLakes":

1. Нажмите кнопку **Start**, выберите пункт **Programs** и, далее, **Microsoft SQL Server 7.0**, а затем укажите команду **Query Analyzer** (Анализатор запросов) для осуществления запроса к таблице `Ramona_PO`.
2. В диалоговом окне **Connect to SQL Server** (Соединение с SQL Server) нажмите кнопку **OK**.
3. Выберите **Ramona** в окне **DB**, а затем введите запрос `SELECT * FROM Ramona_PO`.
4. Нажмите клавишу <F5> для выполнения запроса. Вы увидите элементы записей базы данных `Sku`, `Name`, `Quantity` и `Purchase Number`, относящиеся к покупаемой книге.
5. Выберите базу данных `FiveLakes` в окне **DB**, а затем введите запрос `SELECT * FROM FiveLakes_POReceipt`.
6. Нажмите клавишу <F5> для выполнения запроса. Вы увидите сгенерированный идентификатор продукта (product ID), дату и время, полученное сообщение, принятый алгоритм, а также тип документа отправленного заказа на закупку.

Вопросы для повторения

1. Как называется компонент, который применяется для вызова внешних скриптов?
2. Какие параметры необходимо передавать в процессе работы CIP-конвейера?
3. В каких ступенях CIR-конвейера используются цифровые подписи?
4. Когда компонент `OpenHeader` создает комбинированное сообщение (digest)?

ГЛАВА 5

Создание компонентов конвейера с помощью CIP Manager и работа с ними

Краткий обзор

В этой главе вы изучите программу Commerce Interchange Pipeline Manager (CIP Manager) и узнаете о том, как она настраивается для передачи и приема бизнес-документов электронными партнерами. Вы также ознакомитесь с тем, как упаковывать и передавать бизнес-данные с помощью объектов программы CIP Manager. В заключение, используя Microsoft Visual Basic, создадите компонент конвейера для включения в состав приложения, предназначенный для хранения бизнес-данных в текстовом файле, и включите этот компонент в приложение с помощью CIP Manager.

Так как работа данной программы в основном относится к безопасному обмену бизнес-документами между компьютерами, анализ кода, приведенный в этой главе, а также лабораторные работы построены в расчете на два компьютера, где одному из них назначена роль отправителя (передатчика), а другому — роль получателя документа (приемника). Поэтому установите второй компьютер в соответствии с руководством по установке и присвойте ему уникальное имя, чтобы избежать конфликтов в сети.

Цели и задачи

После прочтения данной главы вы научитесь:

- объяснять, какую роль играет программа CIP Manager в электронной коммерции класса business-to-business;
- передавать и принимать бизнес-документы;

- создавать приложение для передачи объекта бизнес-данных;
- создавать компонент конвейера для включения в состав приложения.

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod05\Mod05Code.txt.

Программа Commerce Interchange Pipeline Manager

В данном разделе рассматриваются:

- краткий обзор CIP Manager;
- передача бизнес-документов.

Краткий обзор CIP Manager

Программа Commerce Interchange Pipeline Manager (CIP Manager) динамически создает и управляет конвейерами, предназначенными для электронной торговли, на основе наборов параметров (profiles) электронных партнеров и их потребностей. Интерфейс CIP Manager предоставляет возможность работы с мастерами (wizards) и страницами свойств (property pages), предназначенными для управления наборами параметров партнера и документа, сохранения усилий, затрачиваемых на ручное редактирование конфигурации

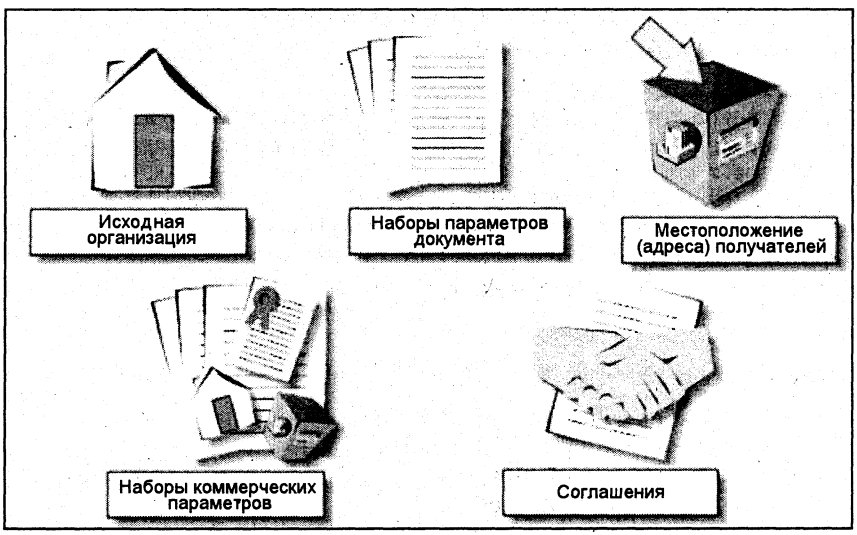


Рис. 5.1. Базовые элементы для CIP Manager

конвейера с помощью программы Pipeline Editor. Программа CIP Manager также предоставляет COM-интерфейсы для того, чтобы разработчики и поставщики сторонних компаний могли создавать совместимые компоненты и легко связывать их вместе в любой необходимой конфигурации.

На рис. 5.1 представлены базовые элементы для CIP Manager, которые следует конфигурировать.

Для того чтобы работать с CIP Manager, необходимо конфигурировать и управлять следующими базовыми элементами:

- исходной организацией (Home organization);
- наборами параметров документов (Document profiles);
- местоположением получателей (Receive locations);
- наборами коммерческих параметров (Trading profiles);
- соглашениями (Agreements).

Исходная организация

Исходная организация (home organization) предоставляет информацию о компании, которая использует CIP Manager. Для создания исходной организации должна быть предоставлена следующая информация:

- наименование организации (компании);
- Web-адрес домашней страницы организации;
- уникальный идентификатор обмена, принадлежащий организации (unique interchange identifier), например, так называемый D&B-номер (D&B D-U-N-S Number, Dun & Bradstreet number[1]) или же номер телефона;

Примечание

D&B D-U-N-S Number представляет собой уникальную идентификационную последовательность из девяти цифр, которые указывают множество программных продуктов и служб (услуг), относящихся к информации о качестве, созданных исключительно корпорацией D&B. D&B D-U-N-S Number является общепринятым в мире идентификатором, используемым в EDI и транзакциях глобальной электронной коммерции. Подробную информацию о вашем D-U-N-S number и его применении смотрите на сайте **http://www.dnb.com**.

- адреса организации для платежей и отгрузки;
- имена электронных партнеров для взаимодействия;
- идентификаторы обмена (interchange identifiers), используемые как передающей, так и принимающей компанией. Когда вы отправляете бизнес-

[1] Dun & Bradstreet Inc. — наиболее широко известное в мире справочное агентство по оценке кредитоспособности и доходности фирм. — *Пер.*

документ, приложение, применяемое в исходной организации (home organization) использует идентификатор обмена для указания электронного партнера, который будет принимать данный документ. Когда ваш электронный партнер принимает документ, его заголовок указывает идентификатор обмена. Этот идентификатор используется электронным партнером для определения, какое приложение должно быть потребителем бизнес-документа. Информация об исходной организации создается и обновляется с помощью Home Information Wizard (Мастера исходной информации).

Наборы параметров документа

Набор параметров документа представляет собой описание бизнес-документа. Для каждого типа принимаемых или передаваемых бизнес-документов необходим отдельный набор параметров. Ниже приведены два типа наборов параметров документа.

- Набор параметров исходного документа (home document profile) описывает документ с точки зрения ваших бизнес-приложений. Такие наборы соответствуют документам, принятым от других организаций (входящие документы), а также документы, которые отправляются в другие организации (исходящие документы). Создание и обновление этих наборов параметров производится с помощью Home Document Profile Wizard (Мастера набора параметров исходного документа).
- Набор параметров документа партнера описывает бизнес-документ с точки зрения бизнес-приложений электронного партнера. Наборы параметров бизнес-документов, полученных электронным партнером, называются наборами параметров входящего документа партнера. Наборы параметров, которые описывают документы, переданные электронным партнером, называются наборами параметров исходящих документов партнера. Эти наборы параметров создаются и обновляются при помощи Partner Document Profile Wizard (Мастера набора параметров документа партнера).

В наборе параметров документа представлена следующая информация:

- идентификатор обмена (interchange identifier), указывающий организацию, ассоциированную с данным набором параметров документа. Наборы параметров исходного документа содержат в себе идентификатор обмена исходной организации (home organization), а наборы параметров документа партнера — идентификатор обмена электронного партнера;
- ссылки на цифровые сертификаты, необходимые для шифрования и цифровой подписи. Эти ссылки являются необязательными (опционными);
- стандартный формат заголовка документа, например, ANSI X12, EDIFACT или CIP XML — указывается в Partner Document Profile Wizard.

Местоположение получателей

Местоположение (адрес) получателей является частью наборов коммерческих параметров (trading profile). Оно указывает, где именно должен быть получен бизнес-документ (например, Web-адрес или же адрес почтового ящика), а также информацию, определяющую способ обработки полученного бизнес-документа. Местоположение получателей также указывает тип используемого транспортного протокола (например, e-mail или HTTP). Программа Receive Location Wizard (Мастер местоположения получателя) создает стандартное (типовое) местоположение, для которого используются общепринятые установки.

Наборы коммерческих параметров

Для упрощения обмена бизнес-документами, осуществляемого с другими электронными партнерами, при помощи CIP Manager вы можете создать и экспортировать набор коммерческих параметров для вашей организации. Этот набор можно опубликовать и сделать доступным для других электронных партнеров.

В наборе коммерческих параметров содержатся следующие сведения:

- информация об исходной организации (Home information);
- наборы параметров документов (Document profiles);
- ннформация о сертификате (Certificate information);
- информация о местоположении (Location information).

Наборы коммерческих параметров записываются с помощью тегов XML. Экспорт набора параметров заключается в получении XML-файла, хранящего набор, от электронного партнера. Создание и обновление наборов коммерческих параметров осуществляется с помощью Export Trading Profile Wizard (Мастера экспорта набора коммерческих параметров).

Соглашения

Соглашения (agreements) описывают отношения между вашими бизнес-документами и электронными документами вашего партнера. Для каждого типа документов, участвующих в обмене, предусмотрено отдельное соглашение. Для создания соглашения вы сначала должны создать или получить набор параметров документа исходной организации (home document profile), а также соответствующий набор параметров документа партнера. Имеются следующие типы соглашений.

- Исходящее соглашение (outgoing agreement) содержит всю информацию, которая нужна программе CIP Manager для генерации CIT-конвейера. Перед созданием исходящего соглашения необходимо сконфигурировать:

- исходный набор параметров (Home profile);
- набор параметров партнера (Partner profile);
- набор параметров исходящих документов исходной организации (Home outgoing document profile);
- набор параметров входящих документов организации партнера (Partner incoming document profile);
- набор параметров передачи (Transmit profile).

❒ Входящее соглашение (incoming agreement) содержит всю информацию, нужную CIP Manager для генерации CIR-конвейера. Перед созданием входящего соглашения необходимо сконфигурировать:

- информацию об исходной организации (Home information);
- информацию о партнере (Partner information);
- набор параметров входящих документов исходной организации (Home incoming document profile);
- набор параметров исходящих документов организации партнера (Partner outgoing document profile).

Передача бизнес-документов

На рис. 5.2 представлены этапы отправки и получения бизнес-документов.

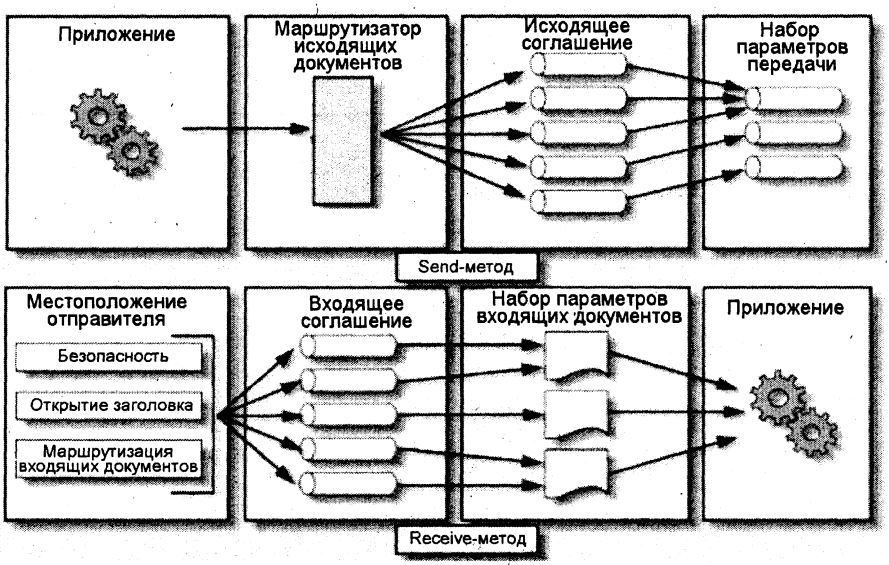


Рис. 5.2. Этапы отправки и получения бизнес-документов

Отправка бизнес-документов

В табл. 5.1 перечислены объекты, используемые программой CIP Manager для обеспечения методов отправки и приема документов.

Таблица 5.1. *Объекты, используемые для отправки и приема документов*

Объект и конвейер	Установки	Использование
`IMSendReceiveTx MtsTxPipeline`	Выполнение (execution): в процессе (in-process) Поддержка TD (передачи документов): Да	Рекомендуется для большинства приложений, включая выполнение внутри другого Т-конвейера, а также при отправке квитанций подтверждения
`IMSendReceive MtsPipeline`	Выполнение (execution): в процессе (in-process) Поддержка TD (передачи документов): Да	Используется в скриптах Microsoft Exchange Server, в которых необходим объект бизнес-данных, но не квитанции подтверждения. Может применяться для любого приложения, в котором не требуются транзакции и квитанции подтверждения
`IMSendReceiveProxy MtsPipeline`	Выполнение (execution): вне процесса (out-of-process) Поддержка TD (передачи документов): Нет	Используется в скриптах Microsoft Exchange Server, где требуются транзакции или подтверждения

Перечисленные объекты используют методы, представленные в табл. 5.2.

Таблица 5.2. *Методы объектов для отправки и приема документов*

Метод	Описание
`Send`	Осуществляет отправку бизнес-документа электронному партнеру
`ReceiveStandard`	Используется в коде или скрипте в стандартном местоположении приема. Осуществляет маршрутизацию бизнес-документа до соответствующего CIR-конвейера
`ReceiveCustom`	Используется в коде или скрипте в заданном местоположении приема. Создает и выполняет CIR-конвейер, связанный с местоположением приема

Подготовка бизнес-документов к передаче

Когда вам нужно отправить бизнес-документ электронному партнеру, приложение вызывает метод `Send` объекта `IMSendReceiveTx` или объекта `IMSendReceiveProxy`. Этот метод определяет, какое исходящее соглашение необходимо использовать. Данное соглашение затем указывает, как осуществлять трансляцию данного документа, сериализацию (serialize) (преобразовывать в последовательную форму), его шифрование, а также указывает транспортный протокол (transport) и заголовок данного документа.

Сериализация

Сериализация (serialization) представляет собой процесс преобразования объекта в поток для дальнейшей передачи. В CIP Manager сериализация выполняется с помощью компонента `MapToXML`. Если передаваемые данные представляют собой текстовый или простой файл, то в CIT-конвейере сериализация не требуется. После приема над сериализованными данными должно быть выполнено обратное преобразование — десериализация (deserialization).

Трансляция

Трансляция (translation) — это процесс преобразования представления данных и имен полей в бизнес-документах для их использования в различных приложениях. Трансляция может выполняться как отправителем, так и получателем при помощи пользовательского компонента, именуемого *механизмом трансляции* (translation engine). Для трансляции используются следующие компоненты, обеспечивающие два способа форматирования бизнес-документов как текста:

- `MapToXML`;
- `Scriptor`.

Порядок выполнения трансляции и сериализации зависит от того, как разработаны данные компоненты. Если механизм трансляции разработан для приема XML-данных, а не объектов, то перед трансляцией данные необходимо конвертировать в XML. И наоборот, если механизм трансляции разработан для чтения и записи объектов, то трансляция должна осуществляться до сериализации. В некоторых случаях обе операции могут быть выполнены одним компонентом.

Передача бизнес-документа

Приложение, осуществляющее отправку, наряду с документом должно иметь так называемый *идентификатор обмена* (interchange identifier), а также имя набора параметров исходящего документа. Все это необходимо методу `Send` для определения, какое *исходящее соглашение* (outgoing agreement) будет

использоваться. Этот процесс именуется *исходящей маршрутизацией* (outgoing routing).

Набор параметров передачи (transmit profile) указывает необходимую степень безопасности, а также транспорт (транспортный протокол), который будет использоваться для отправки бизнес-документа по CIT-конвейеру. Ступень безопасности исходящих документов (Outgoing Security stage) поддерживает компоненты, осуществляющие цифровую подпись и шифрование исходящих бизнес-документов.

Транспортная ступень (Transport stage) осуществляет передачу преобразованного, зашифрованного, снабженного электронной подписью потока байтов, к месту назначения.

Прием бизнес-документов

Когда приложение, осуществляющее отправку, передает бизнес-документ по соответствующему адресу, то скрипт ASP или скрипт Microsoft Exchange по адресу, указанному в адресе приема, создает объект `IMSendReceive`, `IMSendReceiveTx` или `IMSendReceiveProxy` и затем вызывает метод `ReceiveStandard` или же метод `ReceiveCustom`.

Для приема бизнес-документа с помощью CIP Manager требуются следующие элементы:

- стандартный адрес, по которому осуществляется прием входящих бизнес-документов и осуществляется основная (generic) обработка и маршрутизация (routing);
- входящее соглашение (incoming agreement) для выполнения специальной (specific) обработки, в том числе трансляции;
- набор параметров входящего документа, необходимый для указания компонента включения в состав приложения (application integration component);
- система, предназначенная для приема окончательных данных (final data).

В адресе приема представлены ступени, необходимые для указания следующей информации:

- каким образом в ступени безопасности (Security stage) будет осуществляться дешифрация и верификация цифровой подписи;
- как в ступени открытия заголовка (Open Header stage) будет производиться считывание заголовка документа;
- какое входящее соглашение будет использоваться в ступени маршрутизации (Routing stage).

После маршрутизации входящее соглашение вызывает и исполняет соответствующий CIR-конвейер, переданный в бизнес-документе. Входящее

соглашение указывает, какой процесс трансляции и десериализации необходимо использовать, а также, какой компонент включения в состав приложения необходим. Этот компонент передает входящий бизнес-документ в бизнес-приложение, используемое в вашей компании, например в систему бухгалтерского учета.

Анализ кода приложения для передачи объекта бизнес-данных

После того как будут настроены два компьютера для передачи и приема объектов бизнес-данных, вам потребуется приложение, которое будет осуществлять передачу объектов бизнес-данных. Вы можете воспользоваться для этого компонентом третьих фирм, однако, может потребоваться создание пользовательских (custom) компонентов. В лабораторной работе 5 вы познакомитесь с тем, как производится настройка компьютеров для передачи и приема объектов бизнес-данных. В этом разделе вы узнаете о том, как с помощью Visual Basic 6.0 создается приложение, используемое для передачи объекта бизнес-данных.

Для создания такого приложения необходимо выполнить следующие шаги:

1. Создайте, скомпонуйте и заполните объекты `Dictionary` и `SimpleList`.
2. Создайте объект `IMSendReceive` и вызовите метод `Send`.
3. Произведите очистку объектов.

После того как будут выполнены эти шаги, приложение выполнит синтаксический анализ данных, содержащихся в форме, и поместит их в объекты `Dictionary` и `SimpleList`. Затем эти объекты посредством CIT-конвейера отправляются в другое приложение для обработки.

Для того чтобы создать проект на Visual Basic, служащий для передачи объекта бизнес-данных:

1. Запустите Visual Basic 6.0.
2. В окне **New Project** выберите значок **ActiveX DLL**, а затем нажмите кнопку **Open** для подтверждения выбора.
3. В меню **Project** укажите команду **Project1 Properties**. Появится окно **Project1 — Project Properties**.
4. На вкладке **General** выберите значение **Apartment Threaded** в раскрывающемся списке **Threading Model**.
5. Установите флажок **Unattended Execution**.
6. В поле ввода **Project Name** введите `CIPMTransmit`, а затем нажмите кнопку **OK**.

7. В окне **Project Explorer** щелкните правой кнопкой мыши на элементе **Class1**, а затем выберите команду **Save Class1 As**.
8. В окне **Save File As** в поле **File name** введите `CIPMTransmit`, а затем нажмите кнопку **Save**.
9. В меню **View** выберите команду **Properties Window**.
10. На вкладке **Alphabetic** щелкните по строке свойства **MTSTransactionMode**, а затем выберите в раскрывающемся списке значение **3-Uses Transactions**.

Проект `CIPMTransmit` расположен на CD-ROM в папке <папка установки>\DemoCode\Mod05.

Объект бизнес-данных

В этом примере объект бизнес-данных представляет заказ книги, названный `BookOrder`. Этот заказ реализован как объект коммерческого словаря (Commerce Dictionary object), в котором размещается как информация о заказчике, так и список заказанных книг. Перечень книг содержится в элементе `Items` данного словаря. Этот перечень реализован в качестве объекта `SimpleList` сервера коммерции (Commerce Server), в котором любой элемент списка соотнесен с одной книгой. Каждая книга, в свою очередь, представлена объектом `Dictionary` сервера коммерции. В словаре имеется два элемента — `ISBN` и `Title`, служащие для идентификации книги.

Для создания, компоновки (link) и наполнения (populate) объекта бизнес-данных, представляющего заказ книги:

1. Установите ссылку для данного проекта на библиотеку MSCSCore Type Library или файл mscscore.dll. Эта ссылка обеспечивает доступ к классам, необходимым для создания объектов словаря.
2. Объявите и создайте экземпляр нового объекта словаря с именем `BookOrder` с помощью кода:

```
Dim BookOrder As New CDictionary
```

3. После того как объект `BookOrder` будет создан, включите данные о заказчике из формы в заказ на книгу, создав для этого вхождения словаря для каждого элемента, как представлено в следующем фрагменте кода:

```
BookOrder.Customer = txtCustomer.Text
BookOrder.Address = txtAddress.Text
BookOrder.City = txtCity.Text
BookOrder.State = txtState.Text
BookOrder.Zip= txtZip.Text
```

4. Объявите и создайте экземпляр объекта `SimpleList` под именем `BookList` с помощью представленого ниже кода:

```
Dim BookList As New CSimpleList
```

Этот объект будет содержать список книг, выбранных пользователем (заказчиком).

5. Присвойте данному элементу словаря имя `Items` и включите список элементов в заказ на книгу:

```
Set BookOrder.Items = BookList
```

6. Объявите и создайте экземпляр объекта `Dictionary` под именем `Item` для представления каждой книги, с помощью кода, представленного ниже. Вы будете использовать этот объект циклически для каждого элемента (наименования товара) в вашей покупательской корзине.

```
Dim Item As New CDictionary
```

7. Включите данные о каждом из элементов в словарь (Item Dictionary) путем введения в него соответствующих вхождений. Присвойте этим вхождениям имена `ISBN` и `Title`, соответственно.

```
Item.ISBN = strISBN
Item.Title = strTitle
```

8. Включите словарь элементов (item dictionary) в объект списка элементов при помощи вызова метода `Add()` объекта `SimpleList`. Передайте в словарь элементов (item dictionary). После этого выражения закройте цикл. Для передачи элемента в словарь в качестве параметра используется следующее выражение:

```
Call BookList.Add (Item)
```

При выполнении этого выражения осуществляется создание, компоновка и наполнение заказа книги. Этот заказ наряду с информацией о заказчике содержит также список книг.

После создания и наполнения объекта `BookOrder` его необходимо передать.

Для того чтобы осуществить передачу заказа на книгу:

1. Установите ссылки в данном проекте на библиотеку SendReceive Library программы Microsoft CIP Manager 1.0 или на файл ImSndRcv.dll. Эта ссылка обеспечивает доступ к классам, служащим для создания объекта `IMSendReceiveTx`.
2. Для передачи объекта `BookOrder` объявите и создайте экземпляр объекта транспортного словаря, используя следующий код:

```
Dim dictTransport As New CDictionary
```

3. С помощью приведенного ниже кода объявите и создайте экземпляр объекта `IMSendReceiveTx`, чтобы обеспечить возможность вызова метода `Send`:

```
Dim objSendRecv As New ImSendReceiveTx
```

4. По соглашению, когда объект бизнес-данных передан, транспортный словарь содержит вхождение с именем `Object`, которое и является объек-

том бизнес-данных. Здесь объектом бизнес-данных является словарь BookOrder. Включите этот объект в транспортный словарь следующим образом:

```
Set dictTransport.Object = BookOrder
```

5. Для передачи объекта бизнес-данных вызовите метод Send созданного вами объекта IMSendReceiveTx:

```
objSendRecv.Send "Dun & Bradstreet", _
  "123456789","TestSend",dictTransport,2
```

Примечание

Параметр 1 блокирует регистрацию конвейера, 2 — разрешает регистрацию конвейера.

Компонент конвейера для включения в состав приложения

В данном разделе:

- создается компонент конвейера;
- создаются и подключаются страницы свойств компонента (property pages for a component);
- регистрируются признаки принадлежности (affinity) компонента.

После установки CI-конвейера для каждого приложения, которое осуществляет прием данных, потребуется компонент, способный осуществлять "подключение" к CIR-конвейеру.

Вы создадите *компонент для включения в состав приложения* (application integration component). Этот компонент будет осуществлять считывание объекта бизнес-данных, отправленного в конвейер. Затем он начнет осуществлять запись этих данных в файл.

После работы над данным разделом вы сможете:

- создать компонент конвейера;
- соединить конвейер и UI-компоненты (компоненты пользовательского интерфейса);
- зарегистрировать признаки принадлежности компонента.

Программный код для компонента конвейера

Ступень конвейера для включения в состав приложения (application integration stage of a pipeline) предназначена для содержания компонентов, осуществляющих преобразование данных, полученных из объектов бизнес-данных

в формат, который воспринимается конечным приложением. Приложение, которое вы создадите чуть позже, будет осуществлять считывание данных из объекта `dictTransport` и записывать их в текстовый файл.

Вы научитесь:

- создавать проект ActiveX DLL;
- настраивать компонент конвейера;
- программировать интерфейс конвейера;
- включать свойства;
- осуществлять реализацию методов интерфейса `IPipelineComponent`;
- создавать компонент.

Примечание

Выражения, представленные в процедурах, справедливы для осуществления записи данных в текстовый файл. Выражения, указывающие на изменения, которые необходимо сделать в программном коде, чтобы осуществлялась запись в базу данных, приводятся отдельно.

Проект ActiveX DLL

Первым шагом на пути создания компонента является создание проекта ActiveX DLL, в который вы включите программный код, необходимый для реализации компонента.

Для создания ActiveX DLL:

1. Запустите Visual Basic 6.0.
2. В окне **New Project** укажите значок **ActiveX DLL**, а затем нажмите кнопку **Open** для подтверждения выбора.
3. В меню **Project** выберите команду **Project1 Properties**. Появится окно **Project1 — Project Properties**.
4. На вкладке **General** выберите значение **Apartment Threaded** в раскрывающемся списке **Threading Model**.
5. Установите флажок **Unattended Execution**.
6. В поле **Project Name** введите `CIPMReceive`, а затем нажмите кнопку **OK**.
7. В окне **Project Explorer** щелкните правой кнопкой мыши на элементе **Class1**, а затем выберите команду **Save Class1 As**.
8. В окне **Save File As** введите `CIPMRcv` в поле **File name**, а затем нажмите кнопку **Save**.
9. В меню **View** выберите команду **Properties Window**.
10. На вкладке **Alphabetic** щелкните по строке свойства **MTSTransactionMode**, а затем выберите в раскрывающемся списке значение **3-Uses Transactions**.

Теперь вы создали проект ActiveX DLL, который участвует в MTS-транзакциях в `MtsTxPipeline`. Опция `Unattended Execution` обеспечивает такой режим, при котором компонент осуществляет запись сообщений об ошибках в журнал событий (event log), а не отображает их в окнах сообщений.

Необходимый интерфейс

Теперь, когда вы уже создали ActiveX DLL, вам необходимо сделать так, чтобы библиотека Pipeline 1.0 Type Library была доступной для вашего проекта. Это нужно для того, чтобы делать ссылки на объекты и интерфейсы. В каждом компоненте конвейера должна быть предусмотрена реализация интерфейса `IPipelineComponent`, осуществляемая путем передачи указателя интерфейса объекта `OrderForm` между компонентами. Если какой-либо компонент возвращает ошибку, то работа конвейера завершается, и сообщение об ошибке возвращается в вызывающую программу (caller).

Методы интерфейса `IPipelineComponent` указаны в табл. 5.3.

Таблица 5.3. *Методы интерфейса* `IPipelineComponent`

Метод	Описание
`Execute`	Точка входа, с которой OP-конвейер (OPP) или CI-конвейер (CIP) начинает выполнение компонента
`EnableDesign`	Метод, который устанавливает режим дизайна компонента

Для реализации необходимых интерфейсов `IPipelineComponent` в модуле класса проекта ActiveX DLL введите следующий код:

```
Option Explicit
Implements IPipelineComponent
```

Данный компонент осуществляет поддержку одного свойства `Filename`, указывающего путь и имя файла, используемого при записи данных заказа на диск. Вам необходимо создать переменные для хранения имени файла и признака, указывающего, изменялись ли свойства или нет.

Для создания переменных для имени файла и признака включите следующий фрагмент кода:

```
Dim strFilename As String
Dim fDirty As Boolean
```

Для баз данных (databases) объявите переменные, служащие для хранения имени DSN и имени таблицы. Указанная ранее переменная `strFilename` не требуется. Вместо предыдущего фрагмента кода используйте следующий:

```
Dim strDsnName As String
Dim strTableName As String
```

Реализация методов интерфейса *IPipelineComponent*

Далее вам необходимо указать режим функционирования компонента. Имеется два режима: *режим разработки* (design mode) и *режим выполнения* (execution mode). Для подготовки компонента к любому из них используется метод `EnableDesign`. Режим выполнения является режимом, заданным по умолчанию при запуске конвейера.

Для указания режима функционирования компонента следует включить следующий код:

```
Private Sub IPipelineComponent_EnableDesign(ByVal fEnable As Long)
End Sub
```

Когда конвейер создает экземпляр компонента, он запрашивает указатель интерфейса, указывающий на соответствующую реализацию интерфейса `IPipelineComponent` для этого компонента, а затем использует этот указатель для вызова реализации в режиме выполнения.

Запись данных в текстовый файл

Поскольку происходит вызов реализации компонента в режиме выполнения, в ActiveX DLL необходимо добавить код для конвертирования данных из объекта `dicTransport`.

Для записи данных объекта `dictTransport` в текстовый файл:

1. Создайте закрытую (private) функцию, добавив следующий код в файл ActiveX DLL:

```
Private Function IPipelineComponent_Execute _
  (ByVal dictTransport As Object, _
   ByVal pdispContext As Object, ByVal lFlags As Long) As Long
End Function
```

2. Для инициализации необходимых переменных добавьте следующий фрагмент кода:

```
Dim BookOrder
Dim Item
Dim I
```

3. Для перехвата ошибок включите код:

```
On Error GoTo Error
```

Примечание

Код, следуемый за меткой `Error`, должен быть расположен в конце функции.

```
Error:
  IpipelineComponent_Execute = 3
```

4. После выражения `On Error GoTo` поместите следующий код для того, чтобы передать данные, содержащиеся в объекте `dictTransport` в переменную `BookOrder`:

```
Set BookOrder = dictTransport.object
```

5. Для проверки и обработки ситуации, когда поле имени файла пустое, добавьте код:

```
If strFilename = "" Then
  GoTo Error
End If
```

Для баз данных замените условный оператор `If` на следующий:

```
If strDsnName = "" Or strTableName = "" Then
```

6. Для создания текстового файла и записи в него данных добавьте код:

```
Open strFilename For Output As #1
Write #1, "customer Details"
Write #1, BookOrder.Customer
Write #1, BookOrder.Address
Write #1, BookOrder.City & ", " & BookOrder.State & " " & _
          BookOrder.Zip
Write #1, "BookOrder Details"
Write #1, "Number of items =" & Str(BookOrder.Items.Count)
For Each Item In BookOrder.Items
  Write #1, Item.ISBN & ", " & Item.Title
Next
Close #1
```

Для баз данных установите ссылку на библиотеку Microsoft ActiveX Data Objects 2.1 Type Library и замените предыдущий фрагмент кода на представленный ниже:

```
Dim Con As New ADODB.Connection
Dim Rs As New ADODB.Recordset
Dim Po_no As Integer
Con.Open strDsnName, "sa"
Rs.Open strTableName, 2, 2
If Rs.EOF Then
  Po_no = 1
Else
  Rs.MoveLast
  Po_no = Rs("po_no")
  Po_no = Po_no + 1
End If
```

```
Rs.AddNew
Rs("po_no") = Po_no
Rs("customer") = BookOrder.Customer
Rs("address") = BookOrder.Address
Rs("city") = BookOrder.City
Rs("state") = BookOrder.State
Rs("Zip") = BookOrder.Zip
Rs("total_items") = Str(BookOrder.Items.Count)
Rs.Update
Rs.Close
' PO_Details содержит подробную информацию
' о каждой произведенной закупке
Rs.Open "PO_Details", 2, 2
For Each Item In BookOrder.Items
  Rs.AddNew
  Rs("po_no") = Po_no
  Rs("isbn") = Item.ISBN
  Rs("title") = Item.Title
  Rs.Update
Next
Rs.Close
Con.Close
```

7. Для обхода реакции на ошибку добавьте следующий фрагмент кода:

```
IpipelineComponent_Execute = 1
Exit Function
```

Мы продолжим рассмотрение этого кода при изучении подключения страниц свойств (property pages) к данному компоненту.

Программный код для создания и подключения страницы свойств к компоненту

Когда для компонента предусмотрены страницы свойств, пользователи могут указывать имя и путь файла, в который компонент будет осуществлять запись бизнес-данных. Для того чтобы пользователь мог настраивать данный компонент, имеется возможность отображения страницы свойств с помощью другого компонента — так называемого *компонента администрирования конвейера* (pipeline administration component). Страница свойств может быть реализована на Visual Basic в виде формы. Она представляет собой отдельный компонент, который способен отображать пользовательский интерфейс в то время, как для компонента конвейера задан автоматический режим выполнения (unattended execution).

Примечание

Выражения, представленные в процедурах справедливы для осуществления записи данных в текстовый файл. Выражения, указывающие на изменения, которые необходимо сделать в программном коде, чтобы осуществлялась запись в базу данных, приводятся отдельно.

Страницы свойств

С целью создания страниц свойств для данного компонента запустите новый проект в Visual Basic и создайте ActiveX DLL.

Для создания ActiveX DLL:

1. В меню **File** выберите команду **New Project**.
2. В окне **New Project** укажите значок **ActiveX DLL**, а затем нажмите кнопку **OK** для подтверждения выбора.
3. В меню **Project** выберите команду **Project1 Properties**.
4. Появится окно **Project1 — Project Properties**.
5. В поле **Project Name** введите `CIPMReceiveUI`, а затем нажмите кнопку **OK**.
6. В окне **Project Explorer** щелкните правой кнопкой мыши на элементе **Class1**.
7. Выберите команду **Save Class1 As**.
8. В поле **File name** введите `ComponentUI`, а затем нажмите кнопку **Save**.

Ссылки, указывающие на конвейер

Для выбора ссылок на данный конвейер:

1. В меню **Project** укажите команду **References**.
2. В окне **References — CIPMReceiveUI** нажмите кнопку **Browse**.
3. Укажите файл Microsoft Site Server\Bin\MscsCore.dll, а затем нажмите кнопку **Open**.
4. Нажмите кнопку **Browse**, укажите файл Microsoft SiteServer\SiteServer \Commerce\SDK\Commerce\lib\i386\pipecomplib.tlb, нажмите кнопку **Open**, а затем — кнопку **OK**.

Создание интерфейса

После создания ActiveX DLL вам необходимо создать такую форму, которую могли бы видеть пользователи компонента. В данном случае потребуется считывать информацию, вводимую пользователем в текстовом поле. Для отображения пользовательского интерфейса в компоненте должен быть реализован интерфейс `IPipelineComponentUI`.

Для создания пользовательского интерфейса:

1. Добавьте форму в проект и установите значение свойства формы **Name** равным `ComponentUIForm`.
2. Поместите на форме элемент **Label** и для свойства **Caption** введите значение `Filename`.
3. Разместите на форме элемент **TextBox** и установите значение его свойства **Name** равным `txtFilename`.
4. Поместите на форме элемент **CommandButton** и установите значение его свойств **Name** и **Caption** равными `fOK` и `OK`, соответственно.

Для реализации интерфейса `IPipelineComponentUI` введите следующий код в ActiveX DLL:

```
Option Explicit
Implements IPipelineComponentUI
```

Метод `ShowProperties` компонента `IPipelineComponentUI` служит для отображения пользовательского интерфейса.

Пользовательский класс

Чтобы создать пользовательский класс:

1. Для объявления процедуры, используемой при вызове метода `ShowProperties`, включите следующий фрагмент кода:

```
Private Sub IPipelineComponentUI_ShowProperties _
        (ByVal pdispComponent As Object)
End Sub
```

2. Для захвата и обработки ошибок добавьте код:

```
On Error GoTo HandleError
HandleError:
MsgBox Err.Description & " " & Err.Number
```

3. Для организации взаимодействия между пользовательским интерфейсом и компонентом добавьте следующий код:

```
Dim pdispPCA As IPipelineComponentAdmin
```

4. Для заполнения словаря конфигурацией компонента добавьте:

```
Set pdispPCA = pdispComponent
Dim dictConfig As Cdictionary
Set dictConfig = pdispPCA.GetConfigData
```

5. Для создания формы, предназначенной для считывания данных, указанных пользователем, введите следующее:

```
Dim frm As New ComponentUIForm
frm.txtFilename.Text = dictConfig("filename")
frm.Show 1
If frm.fOK Then
  dictConfig("filename") = frm.txtFilename.Text
  pdispPCA.SetConfigData dictConfig
End If
```

Для баз данных замените предыдущий фрагмент кода представленным ниже:

```
Dim frm As New ComponentUIForm
frm.txtDSNname.Text = dictConfig("dsnname")
frm.txtTablename.Text = dictConfig("tablename")
frm.Show 1
If frm.fOK Then
  dictConfig("dsnname") = frm.txtDSNname.Text
  dictConfig("tablename") = frm.txtTablename.Text
  pdispPCA.SetConfigData dictConfig
End If
```

Пользователю понадобится форма, чтобы ввести путь и имя файла, в котором будет храниться бизнес-документ.

6. Для освобождения памяти, занимаемой переменными, и выхода из процедуры добавьте код:

```
Set pdispPCA = Nothing
Set dictConfig = Nothing
Exit Sub
```

Теперь вы создали компонент конвейера и дополнительный компонент для реализации пользовательского интерфейса, предназначенный для редактирования свойств компонента. Однако эти два компонента еще не связаны (не скомпонованы). К тому же компонент конвейера не будет отображаться в выпадающих списках программы CIP Manager. Для того чтобы это выполнялось, необходимо, чтобы данный компонент поддерживал некоторые интерфейсы. В следующем разделе вы изучите методы, используемые для интеграции этих двух компонентов.

Подключение страницы свойств к компоненту

Сначала откройте проект ActiveX DLL, который вы создали для построения страниц свойств. Для того чтобы подключить страницы свойств к данному компоненту, добавьте к проекту ссылки на следующие библиотеки:

- Commerce — MSCSCore Type Library;
- Microsoft Commerce Pipeline Components Type Library.

Для обеспечения необходимой функциональности данный компонент должен поддерживать дополнительные интерфейсы.

Примечание

Если имена указанных библиотек отсутствуют в списке, отображаемом в диалоговом окне **References**, нажмите кнопку **Browse**, найдите mscscore.dll и pipecomplib.tlb и включите ссылки на них в проект.

Для реализации классов, необходимых для интерфейсов:

1. Найдите в файле следующую строку кода:

```
Implements IPipelineComponentUI
```

2. Для реализации интерфейсов добавьте следующее:

```
Implements ISpecifyPipelineComponentUI
Implements IPipelineComponentDescription
Implements IPipelineComponentAdmin
Implements IPersistDictionary
```

Для того чтобы dll-файл, который реализует пользовательский интерфейс для редактирования в среде Pipeline Editor или CIP Manager, возвращал программный идентификатор (progID), вам потребуется интерфейс `IspecifyPipelineComponentUI`.

Для реализации интерфейса `ISpecifyPipelineComponentUI` в конце файла добавьте следующий код, необходимый для создания требуемой функции:

```
Private Function _
  ISpecifyPipelineComponentUI_GetPipelineComponentUIProgID() As String
  ISpecifyPipelineComponentUI_GetPipelineComponentUIProgID = _
    "CIPMReceiveUI.ComponentUI"
End Function
```

Метод `GetPipelineComponentUIProgID` возвращает программный идентификатор ассоциированного компонента, обеспечивающего пользовательский интерфейс для данного компонента. Этот метод разрешает включение вызывающего приложения, например, программы CIP Manager, для того, чтобы определить, какой пользовательский интерфейс связан с данным компонентом. В нашем случае таким интерфейсным компонентом является `CIPMReceiveUI.ComponentUI`.

Интерфейс `IPipelineComponentDescription` позволяет компонентам конвейера публиковать значения, которые они считывают из контекста конвейера, а также пары "имя/значение", которые они считывают из транспортного словаря или же записывают в него. В рассматриваемом случае этот компонент не считывает ничего из контекстного словаря. Напротив, он считывает рабочие данные из транспортного словаря, но не производит обратной записи в транспортный словарь.

Для реализации интерфейса IPipelineComponentDescription:

1. Метод ContextValuesRead идентифицирует элементы контекста конвейера, считываемые компонентом. Для возможности вызова этого метода ContextValuesRead в конец файла добавьте следующее:

```
Private Function _
  IPipelineComponentDescription_ContextValuesRead() As Variant
  IPipelineComponentDescription_ContextValuesRead = Array("None")
End Function
```

   Метод ValuesRead идентифицирует пары "имя/значение", которые компонент считывает из формы OrderForm.

2. В конец файла добавьте представленный ниже код для реализации метода ValuesRead:

```
Private Function IPipelineComponentDescription_ValuesRead() _
        As Variant
  IPipelineComponentDescription_ValuesRead = Array("working_data")
End Function
```

   Метод ValuesWritten идентифицирует пары "имя/значения", которые компонент модифицировал в форме OrderForm.

3. Для реализации метода ValuesWritten добавьте в конец файла следующее:

```
Private Function _
  IPipelineComponentDescription_ValuesWritten() As Variant
  IPipelineComponentDescription_ValuesWritten = Array("None")
End Function
```

Вызов методов GetConfigData и SetConfigData компонента IPipelineComponentAdmin осуществляется в CIPMReceiveUI.ComponentUI. Метод GetConfigData позволяет компоненту пользовательского интерфейса считывать текущее значение из компонента и изначально отображать его на странице свойств. Метод SetConfigData осуществляет запись обновленного значения из страницы свойств в компонент, когда пользователь изменяет какое-либо значение и щелкает мышью по кнопке **OK**.

Для реализации компонента IPipelineComponentAdmin:

1. Чтобы иметь возможность вызывать метод GetConfigdata, в конце файла добавьте следующее:

```
Private Function IPipelineComponentAdmin_GetConfigData() _
         As Object
  Dim objectConfig As New Cdictionary
  objectConfig.Value("filename") = strFilename
  Set IPipelineComponentAdmin_GetConfigData = objectConfig
End Function
```

Для баз данных замените предыдущий код следующим:

```
Private Function IPipelineComponentAdmin_GetConfigData() _
        As Object
  Dim objectConfig As New Cdictionary
  objectConfig.Value("dsnname") = strDsnName
  objectConfig.Value("tablename") = strTableName
  Set IPipelineComponentAdmin_GetConfigData = objectConfig
End Function
```

2. Для реализации метода `SetConfigData` в конце файла добавьте следующее:

```
Private Sub IPipelineComponentAdmin_SetConfigData _
  (ByVal pDict As Object)
  strFilename = pDict.Value("filename")
End Sub
```

Для баз данных замените предыдущий код представленным ниже:

```
Private Sub IPipelineComponentAdmin_SetConfigData _
  (ByVal pDict As Object)
  strDsnName = pDict.Value("dsnname")
  strTableName = pDict.Value("tablename")
End Sub
```

Если бы вы сейчас стали создавать проект, то данный компонент появился бы в CIP Manager в списках компонентов, а страница свойств отображалась бы корректно. Однако CIP Manager не сможет "вспомнить" значение свойства, если оно было отредактировано в странице свойств.

Для решения данной проблемы вам необходимо обеспечить поддержку интерфейса `IPersistDictionary`.

Метод `GetProgID` используется для указания используемого компонента.

Для реализации метода `GetProgID` в конце файла добавьте следующее:

```
Private Function IPersistDictionary_GetProgID() As String
  IPersistDictionary_GetProgID = "CIPMReceive.CIPMRcv"
End Function
```

Метод `InitNew` используется для инициализации значений свойства.

Для реализации метода `InitNew` в конце файла добавьте код:

```
Private Sub IPersistDictionary_InitNew()
  strFilename = ""
  fDirty = False
End Sub
```

Для баз данных замените предыдущий код таким фрагментом:

```
Private Sub IPersistDictionary_InitNew()
  strDsnName = ""
  strTableName = ""
  fDirty = False
End Sub
```

Метод `IsDirty` используется для указания, что данное свойство было изменено.

Для реализации метода `IsDirty` в конце файла добавьте:

```
Private Function IPersistDictionary_IsDirty() As Long
  IPersistDictionary_IsDirty = fDirty
End Function
```

Метод `Save`, относящийся к `IPersistDictionary`, копирует значения свойств компонента в словарь, который затем записывается на диск.

Для реализации метода `Save` в конце файла добавьте код:

```
Private Sub IPersistDictionary_Save _
  (ByVal pdispDict As Object, ByVal fSameAsLoad As Long)
  pdispDict.Filenm = strFilename
End Sub
```

Для баз данных замените предыдущий код таким:

```
Private Sub IPersistDictionary_Save _
  (ByVal pdispDict As Object, ByVal fSameAsLoad As Long)
  pdispDict.Dsnnm = strDsnName
  pdispDict.Tablenm = strTableName
End Sub
```

Метод `Load` принимает значение свойств компонента из существующего словаря и копирует эти значения в свойства компонента.

Для реализации метода `Load` в конец файла добавьте представленный ниже код:

```
Private Sub IPersistDictionary_Load(ByVal pdispDict As Object)
  strFilename = pdispDict.Filenm
End Sub
```

Для баз данных замените предыдущий код:

```
Private Sub IPersistDictionary_Load(ByVal pdispDict As Object)
  strDsnName = pdispDict.Dsnnm
  strTableName = pdispDict.Tablenm
End Sub
```

Теперь у компонента имеются все интерфейсы, необходимые для того, чтобы его можно было использовать в ступени включения в состав приложения (Application Integration stage) в CIP Manager и Pipeline Editor. Однако для того, чтобы компонент был включен в списки в программах CIP Manager и Pipeline Editor, его необходимо зарегистрировать и создать для него так называемые *признаки принадлежности* (affinity) на том компьютере, где он будет использоваться.

Регистрация признаков принадлежности компонента

Для того чтобы можно было использовать компонент в конвейере, вам необходимо, прежде всего, зарегистрировать его с указанием идентификаторов (ID) категорий, предусмотренных для всех компонентов конвейера, а также для указания конкретной ступени, в которой компонент будет использоваться.

При создании компонента следует устанавливать свойства проекта для обеспечения совместимости на двоичном уровне. Данный режим создает гарантии того, что при каждой новой компиляции dll-файла, не будет создаваться новый *глобальный уникальный идентификатор* (GUID, Globally Unique Identifier) для данного объекта.

Назначение признаков принадлежности

Каждый компонент конвейера должен быть сопоставлен с двумя идентификаторами категорий. Один идентификатор категории (category ID) однозначно определяет данный компонент как компонент конвейера. Другой идентификатор категории, именуемый *признаком принадлежности* (affinity), указывает ступень, в которой предполагается использовать данный компонент.

Чтобы сделать эти сопоставления, необходимо выполнить представленную ниже процедуру на том компьютере, на котором вы используете созданный вами пользовательский компонент (т. е. на компьютере-приемнике). Для выполнения этих сопоставлений будет использоваться файл с расширением reg.

- Первая группа строк reg-файла обеспечивает появление компонента в списках компонентов при выборе пункта меню **Show All**. Вторая группа строк обеспечивает появление компонента в списке компонентов, относящихся к конкретной ступени.
- Первая и вторая строки служат для установки идентификатора класса созданного компонента.
- Идентификаторы категории перечисляются в файле pipe_stages.h, имеющимся в Commerce Server SDK, который можно найти по следующему адресу: Microsoft Site Server\Commerce\SDK\Commerce\include\pipe_stages.h.

- Установите идентификатор категории ступени, в которой предполагается использовать компонент конвейера. Значение идентификатора категории отображается в комментарии компонента. Скопируйте и вставьте это значение в reg-файл, замещая значение `catid_stage`. В данном случае, поскольку вы создаете компонент включения в состав приложения (application integration component), необходимо использовать идентификатор категории для ступени включения в приложение (application integration stage), который имеет следующее значение: `{D3A71C37-FFC7-11D0-B885-00C04FD7A0F9}`.

  В reg-файле будут следующие пункты:

```
[HKEY_CLASSES_ROOT\CLSID\{ clsid_component}\Implemented
Categories\{CF7536D0-43C5-11D0-B85D-00C04FD7A0FA}]
[HKEY_CLASSES_ROOT\CLSID\{ clsid_component}\Implemented
Categories\{ catid_stage}]
```

  Эти пункты имеются в файле affinity.reg, который вы будете использовать в лабораторной работе. Для получения значения `clsid_component` можно воспользоваться программой Registry Editor или OLE Viewer.

- Анализируя пункты реестра, обратите внимание на следующее:
  - идентификатором класса данного компонента является `clsid_component`;
  - идентификатором категории ступени, в которой будет использоваться данный компонент, является `catid_stage`;
  - идентификатором категории компонента, который идентифицирует этот объект как компонент конвейера, является `CF7536D0-43C5-11D0-B85D-00C04FD7A0FA`.

- Наконец, для обновления реестра щелкните правой кнопкой мыши на reg-файле, и затем выберите команду **Merge** (Слияние). Если слияние было осуществлено успешно, то соответствующее сообщение укажет, что информация включена в реестр.

После успешной регистрации компонента, его можно использовать в Pipeline Editor и CIP Manager.

Для использования созданных компонентов:

1. На компьютере-приемнике укажите новый компонент в качестве компонента включения в состав приложения (application integration component).
2. В наборе параметров входящего документа для вашего компонента выберите **Application integration component**.
3. Установите свойства данного компонента с помощью созданной для него страницы свойств.

4. Сконфигурируйте соглашения на обоих компьютерах для обеспечения поддержки сериализации.

Этот шаг необходим, поскольку данное приложение передает объект бизнес-данных, а не только текст. Для того чтобы отправить объект бизнес-данных, объект должен быть включен в транспортный словарь как объект с именем `Object`, и, кроме того, в CIT- и CIR-конвейеры должны быть включены компоненты `MapToXML` и `MapFromXML`, соответственно.

Лабораторная работа 5. Формирование и использование компонента для включения в состав приложения в программе CIP Manager

Цели

После выполнения этой лабораторной работы вы должны уметь:

- пользоваться программой CIP Manager для настройки компьютеров, необходимой для организации передачи и приема бизнес-документов между электронными партнерами;
- создавать компонент для включения в состав приложения (application integration pipeline component) в Microsoft Visual Basic;
- создавать признаки принадлежности (affinity).

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь практические навыки работы с Visual Basic 6.0;
- понимать, в чем заключается различие между рабочей группой (workgroup) и доменом (domain).

Подготовка к лабораторной работе

Для выполнения данной работы вам потребуется два компьютера: один — для передачи бизнес-документов, а другой — для приема. Оба компьютера должны иметь одинаковую конфигурацию аппаратного и программного обеспечения. Если у вас нет такой возможности, постарайтесь выполнить данную работу до того пункта, где требуется импортировать данные со второй машины.

Перейдите в каталог <папка_установки>\Labs\Lab05\StartCode и зарегистрируйте CIPMReceiveUI.dll, введя Regsvr32 <папка_установки>\Labs\Lab05\StartCode\CIPMReceiveUI.dll в поле **Open**, расположенном в окне **Run**.

Примечание

Время, необходимое для выполнения данной работы, — 45 минут.

Упражнение 1. Настройка компьютеров

В данном упражнении для настройки компьютеров с целью передачи и приема бизнес-документов вам понадобится программа CIP Manager.

Компьютер-приемник

Для создания Web-адреса на принимающем компьютере в Microsoft Windows NT Explorer в папке Inetpub\wwwroot\ создайте новую папку CIPM.

По умолчанию, для этой папки имеется разрешение на выполнение asp-страниц.

Для запуска CIP Manager в меню кнопки **Start** укажите пункт **Programs**, далее выберите последовательно команды **Microsoft Site Server** и **Commerce**, а затем щелкните на **CIP Manager**.

Для настройки информации об исходной организации на компьютере-приемнике:

1. В управляющем дереве (console tree) щелкните правой кнопкой мыши на элементе **CIP Manager**, а затем выберите команду **New Home Organization**.
2. В первом окне **Welcome page** нажмите кнопку **Next**.
3. В окне **Organization Name and URL** (Наименование и URL организации) в поле **Home organization name** (Наименование исходной организации) введите `Ramona`, а затем нажмите кнопку **Next**.
4. В окне **Interchange Identifier** выберите **Use Dun & Bradstreet number to identify my organization**, введите `123456789` (девять цифр) в качестве идентификатора обмена в поле **D-U-N-S Number**, а затем щелкните по кнопке **Next**.
5. Оставьте незаполненными поля во всех остальных окнах, служащих для указания адреса и другой контактной информации.
6. Нажмите кнопку **Finish** для завершения работы Home Information Wizard (Мастера информации об исходной организации).

Для создания набора параметров входящих документов на компьютере-приемнике:

1. В управляющем дереве последовательно раскройте **CIP Manager**, **CIP Manager (local)**, **Ramona**, щелкните правой кнопкой мыши на элементе **Document Profiles**, а затем выберите команду **New Document Profile**.
2. В первом окне нажмите кнопку **Next**.
3. В окне **Incoming or Outgoing Document** (Входящие или исходящие документы) выберите **Create a new incoming document profile** (Создать новый набор параметров входящих документов), а затем щелкните по кнопке **Next**.
4. В окне **Incoming Document Profile Name** (Имя набора параметров входящих документов) в поле **Incoming document profile name** введите `FiveLakesPO`, а затем нажмите кнопку **Next**.
5. В окне **Document Header Type** (Тип заголовка документа) выберите **CIP XML** в области **Document header type**, а затем нажмите кнопку **Next**.
6. В окне **Encryption Certificate** (Сертификат шифра) проверьте, чтобы опция **Sending partner will encrypt this document, using this certificate** (Партнер-отправитель будет осуществлять шифрование данного документа, используя данный сертификат) была отключена, а затем щелкните по кнопке **Next**.
7. В окне **Application Integration Component** (Компонент включения в состав приложения) выберите **POtoFile (Write toFile)** в области **Application Integration Component**, а затем нажмите кнопку **Next**.
8. Нажмите кнопку **Finish** для завершения работы Home Document Profile Wizard (Мастера набора параметров исходного документа).

Для создания стандартного адреса приема на компьютере-приемнике:

1. В управляющем дереве, отображаемом программой CIP Manager, раскройте соответствующий узел (node), щелкните правой кнопкой мыши на элементе **Receive Locations**, а затем выберите команду **New Receive Location**.
2. В первом окне нажмите кнопку **Next**.
3. В окне **Receive Location Name** (Адрес приема) введите `Seattle` в поле **Receive location name**, а затем нажмите кнопку **Next**.
4. В окне **Receive Location Transport** (Транспорт для приема) выберите **Receive using a Web Server (IIS)** в качестве типа транспорта, а затем щелкните по кнопке **Next**.
5. В поле **Your Internet Address** введите **http://*Studentxx*/CIPM/ReceiveStandard.asp** (где ***Studentxx*** — имя компьютера, настроенного для приема) и убедитесь, чтобы флажок **Copy Receive Standard Script to Web server** был установлен. Щелкните кнопку **Browse**, расположенную рядом с полем **Local file path and name**, а затем перейдите в каталог CIPM, соз-

данный на предыдущем этапе. В качестве имени файла должно быть задано имя ReceiveStandard.asp. Нажмите кнопку **Save**, а затем — кнопку **Next**.

6. Мастер осуществит копирование скрипта ReceiveStandard.asp из папки Microsoft Site Server\Bin\CIP Manager в указанную папку.
7. Нажмите кнопку **Finish**.

Для осуществления экспорта набора коммерческих параметров (trading profile):

1. В управляющем дереве программы CIP Manager щелкните правой кнопкой мыши на элементе **Ramona**.
2. Выберите команду **Export Profile**. В первом окне нажмите кнопку **Next**.
3. Нажимайте кнопку **Next** в окнах **Available Certificates**, **Available Document Profiles** и **Available Receive Locations**.
4. В окне **Export Path and Internet Address** щелкните по кнопке **Browse**.
5. Перейдите в папку Inetpub\wwwroot\CIPM. В поле **File name** измените имя на RamonaProfile.tpp, а затем нажмите кнопку **Save**.
6. В поле **Internet address** введите адрес **http://*Studentxx*/CIPM/RamonaProfile.tpp** (где ***Studentxx*** — имя компьютера, настроенного в качестве приемника) для получения данного набора коммерческих параметров с другого компьютера.
7. Нажмите кнопку **Next**, а затем — кнопку **Finish**.

Примечание

Прежде чем настраивать компьютер-приемник, необходимо настроить компьютер-передатчик.

Компьютер-передатчик

В последующих процедурах вы произведете настройку компьютера для передачи бизнес-документов. Это упражнение невозможно выполнить, используя лишь один компьютер, т. к. на одном и том же компьютере нельзя создать две "локальные" конфигурации. Если у вас есть только один компьютер, откройте файл ReceiveStandard.asp, размещенный в каталоге CIPM, и исследуйте расположенный в нем набор экспортируемых параметров.

Для создания Web-адреса на компьютере-передатчике в Microsoft Windows NT Explorer в папке Inetpub\wwwroot\ создайте новую папку CIPM. По умолчанию, для этой папки разрешено выполнение скриптов asp-страниц.

Для запуска CIP Manager в меню кнопки **Start** укажите пункт **Programs**, далее выберите последовательно команды **Microsoft Site Server** и **Commerce**, а затем щелкните на **CIP Manager**.

Для настройки информации об исходной организации на компьютере-передатчике:

1. В управляющем дереве (console tree) щелкните правой кнопкой мыши на элементе **CIP Manager**, а затем выберите команду **New Home Organization**.
2. В первом окне **Welcome page** щелкните по кнопке **Next**.
3. В окне **Organization Name and URL** (Название и URL организации) в поле **Home organization name** (Наименование исходной организации) введите `FiveLakes`, а затем нажмите кнопку **Next**.
4. В окне **Interchange Identifier** выберите **Use Dun & Bradstreet number toidentify my organization** (Для идентификации моей компании использовать D&B-идентификатор) введите `999999999` (девять цифр) в качестве идентификатора обмена в поле **D-U-N-S Number**, а затем нажмите кнопку **Next**.
5. Оставьте незаполненными поля во всех остальных окнах, служащих для указания адреса и другой контактной информации.
6. Нажмите кнопку **Finish** для завершения работы Home Information Wizard (Мастера информации об исходной организации).

Для создания исходящего соглашения:

1. В управляющем дереве раскройте последовательно **CIP Manager**, **CIP Manager (local)** и **FiveLakes**, щелкните правой кнопкой мыши на элементе **Agreements**, а затем выберите команду **New Outgoing Agreement**.
2. В первом окне щелкните по кнопке **Next**.
3. В окне **Agreement Name** (Наименование соглашения) введите `Ramona` в поле **Outgoing agreement name**, а затем нажмите кнопку **Next**.
4. В окне **Trading Partner** (Торговый партнер) нажмите кнопку **New** для запуска Partner Information Wizard (Мастера информации о партнере).
5. В первом окне мастера Partner Information Wizard щелкните по кнопке **Next**.
6. Выберите **Import the partner from a trading profile**. В поле, расположенном ниже, введите адрес набора коммерческих параметров для компьютера-приемника: **http://*Studentxx*/CIPM/RamonaProfile.tpp** (где ***Studentxx*** — имя компьютера, настроенного для приема), а затем нажмите кнопку **Next**.
7. Нажмите кнопку **Finish** для завершения работы мастера Partner Information Wizard. Мастер Outgoing Agreement Wizard продолжит свою работу.
8. В окне **Trading Partner** нажмите кнопку **Next**. В качестве коммерческого партнера (trading partner) теперь указана компания "Ramona".

9. В окне **Document Profiles** Outgoing Agreement Wizard (Мастера исходящего соглашения) нажмите кнопку **New** для указания набора параметров документа для данного исходящего соглашения. Произойдет запуск мастера Home Document Profile Wizard. Нажмите кнопку **Next**.
10. В окне **Outgoing Document Profile Name** введите `POtoRamona` в поле **Outgoing document profile name**, а затем нажмите кнопку **Next**.
11. В окне **Document Header Type** выберите **CIP XML** в области **Document header type**, а затем щелкните по кнопке **Next**.
12. В окне **Signature Certificate** убедитесь, чтобы флажок **Digitally sign this-document, using this signature certificate** был сброшен. Нажмите кнопку **Next**.
13. Нажмите кнопку **Finish** для завершения работы мастера Home Document Profile Wizard. Мастер Outgoing Agreement Wizard продолжит свою работу.
14. В качестве набора параметров входящих документов коммерческого партнера выберите **FiveLakesPO**. Этот набор параметров документов партнера был импортирован из набора параметров компьютера-приемника. Установите флажок **Translation/Serialization Required**. Проверьте, что компонент `MaptoXML` выбран в раскрывающемся списке, а затем нажмите кнопку **Next**.
15. В окне **Transmit Profile** выберите набор параметров компьютера-приемника, который был импортирован из набора параметров партнера.
16. В окне **Transmit Profile** щелкните по кнопке **Next**, а затем — по кнопке **Finish**.

Для осуществления экспорта набора коммерческих параметров:

1. В управляющем дереве программы CIP Manager щелкните правой кнопкой мыши на элементе **FiveLakes**, а затем выберите команду **Export Profile**. Произойдет запуск Export Trading Profile Wizard (Мастера экспорта набора коммерческих параметров).
2. В первом окне щелкните по кнопке **Next**.
3. Нажимайте кнопку **Next** в окнах **Available Certificates**, **Available Document Profiles** и **Available Receive Locations**.
4. В окне **Export Path and Internet Address** щелкните по кнопке **Browse**.
5. Перейдите в папку Inetpub\wwwroot\CIPM, расположенную на данном компьютере. В поле **File name** замените имя на FiveLakesProfile.tpp, а затем нажмите кнопку **Save**.
6. Введите адрес для приема этих коммерческих параметров с другого компьютера: **http://*Studentxx*/CIPM/FiveLakesProfile.tpp** (где ***Studentxx*** — имя компьютера, настроенного для передачи), а затем нажмите кнопку **Next**.

Данный адрес указывает такой же путь и имя файла, который был указан для экспорта набора параметров.

7. Нажмите кнопку **Finish**.

Завершение настройки компьютера-приемника

Для запуска Incoming Agreement Wizard (Мастера входящего соглашения) на компьютере-приемнике:

1. В управляющем дереве программы раскройте **CIP Manager**, **CIP Manager (local)** и **Ramona**, щелкните правой кнопкой мыши на элементе **Agreements**, а затем выберите команду **New Incoming Agreement**.
2. В первом окне щелкните по кнопке **Next**.
3. В окне **Agreement Name** (Наименование соглашения) введите `FiveLakes` в поле **Incoming agreement name**, а затем нажмите кнопку **Next**.
4. В окне **Trading Partner** (Торговый партнер) нажмите кнопку **New** для запуска Partner Information Wizard (Мастера информации о партнере).
5. В первом окне мастера щелкните по кнопке **Next**.
6. Выберите **Import the partner from a trading profile**. В расположенном ниже поле введите адрес набора коммерческих параметров для компьютера-передатчика: **http://*Studentxx*/CIPM/FiveLakesProfile.tpp** (где ***Studentxx*** — имя компьютера-передатчика), а затем щелкните по кнопке **Next**.
7. Нажмите кнопку **Finish** для завершения работы мастера Partner Information Wizard. Мастер Incoming Agreement Wizard продолжит работу.
8. В окне **Trading Partner** нажмите кнопку **Next**.
9. В окне **Document Profiles** выберите **POtoRamona** в качестве набора параметров исходящих документов партнера, а также **FiveLakesPO** в качестве вашего набора параметров входящих документов. Установите флажок **Translation/Deserialization required**, убедитесь, что компонент `MapFromXML` выбран в раскрывающемся списке, а затем нажмите кнопку **Next**.
10. Обратите внимание, что в окне **Available Receive Locations** отображается ранее заданный адрес приема. Нажмите кнопку **Next**, а затем — кнопку **Finish**.

Упражнение 2. Создание компонента конвейера

Данное упражнение выполняется на компьютере, настроенном для приема бизнес-документа.

В этом упражнении вы создадите компонент конвейера для включения в состав приложения (application integration pipeline component) с помощью Microsoft Visual Basic. Этот компонент будет снабжен пользовательскими страницами параметров, реализованными в отдельном проекте Microsoft

ActiveX DLL. Этот компонент уже был зарегистрирован на вашем компьютере (имя проекта — `CIPMReceiveUI`, а имя класса — `ComponentUI`). Вы также напишете код, предназначенный для чтения свойств компоненты конвейера, которые вы установили с помощью страницы свойств.

Для запуска Microsoft Visual Basic 6.0 нажмите кнопку **Start**, укажите пункт **Programs**, далее выберите команду **Microsoft Visual Studio 6.0**, а затем — **Microsoft Visual Basic 6.0**.

Для того чтобы открыть проект ActiveX DLL:

1. В меню **File** укажите команду **Open Project**.
2. Перейдите в папку <папка_установки>\Labs\Lab05\StartCode, а затем выберите файл CIPMReceive.vbp.

Для настройки компонента:

1. В меню **Project** выберите команду **CIPMReceive Properties**.
2. На вкладке **General** установите значение **Threading Model** в раскрывающемся списке **Apartment Threaded**.
3. В окне дерева проекта щелкните на модуле класса CIPMRcv. В окне **Properties** установите свойство **MTSTransactionMode** равным **3-Uses Transactions**.
4. В меню **File** выберите команду **Save Project As**. В диалоговом окне **Save Project As** перейдите в папку <папка установки>\Labs\Lab05\StartCode, расположенную на вашем компьютере, нажмите кнопку **Save** для сохранения файла проекта[1].

Для включения ссылок в проект:

1. В меню **Project** укажите команду **References**.
2. Установите флажки **pipeline 1.0 Type Library** или **pipeline.dll**, **Commerce — MSCSCore Type Library** или **mscscore.dll** и **Microsoft Commerce Pipeline Components Type Library** или **pipecomplib.tlb**.

Для поддержки интерфейса конвейера и задания свойства:

1. В окне **Project Explorer** щелкните правой кнопкой мыши на модуле класса CIPMRcv, а затем в появившемся меню выберите команду **View Code**.
2. В начале модуля класса введите следующее:

```
Implements IpipelineComponent
Dim strFilename As String
Dim fDirty As Boolean
```

[1] Для сохранения файла класса необходимо в окне **Project Explorer** щелкнуть правой кнопкой мыши на элементе **Class1**, а затем выбрать команду **Save Class1 As**. — *Ред.*

```
Private Sub IPipelineComponent_EnableDesign(ByVal fEnable As Long)
End Sub

Private Function IPipelineComponent_Execute _
    (ByVal dictTransport As Object, ByVal pdispContext As Object, _
     ByVal lFlags As Long) As Long
  Dim I
  Dim BookOrder
  Dim item
  Set BookOrder = dictTransport.object
  On Error GoTo Error
  If strFilename = "" Then
    GoTo Error
  End If
  Open strFilename For Output As #1
  Write #1, "Customer Details"
  Write #1, BookOrder.Customer
  Write #1, BookOrder.Address
  Write #1, BookOrder.City & ", " & BookOrder.State & _
            " " & BookOrder.Zip
  Write #1, "BookOrder Details"
  Write #1, "Number of items =" & Str(BookOrder.Items.Count)
  For Each item In BookOrder.Items
    Write #1, item.ISBN & ", " & item.Title
  Next
  Close #1
  IPipelineComponent_Execute = 1
  Exit Function
Error:
  IPipelineComponent_Execute = 3
End Function
```

Для чтения свойств компонента в конце файла добавьте следующее:

```
Private Function IPersistDictionary_GetProgID() As String
  IPersistDictionary_GetProgID = "CIPMReceive.CIPMRcv"
End Function

Private Sub IPersistDictionary_InitNew()
  strFilename = ""
  fDirty = False
End Sub

Private Function IPersistDictionary_IsDirty() As Long
  IPersistDictionary_IsDirty = fDirty
End Function
```

```
Private Sub IPersistDictionary_Load(ByVal pdispDict As Object)
  strFilename = pdispDict.Filenm
End Sub

Private Sub IPersistDictionary_Save(ByVal pdispDict As _
    Object, ByVal fSameAsLoad As Long)
  pdispDict.Filenm = strFilename
End Sub
```

Для компоновки проекта:

1. В меню **File** выберите команду **Save Project**.
2. В меню **File** укажите команду **Make CIPMReceive.dll**.
3. Сохраните dll-файл в папке <папка_установки>\Labs\Lab05\StartCode.

Упражнение 3. Регистрация признаков принадлежности

В этом упражнении вы зарегистрируете компонент конвейера в ступени включения в состав приложения (Application Integration stage), для которого данный компонент предназначен. Вы назначите и протестируете признаки принадлежности (affinity).

Для редактирования файла affinity.reg в Microsoft Windows NT Explorer перейдите в папку Lab\Lab05\StartCode, содержащую файл Affinity.reg, щелкните правой кнопкой мыши на файле Affinity.reg, а затем выберите команду **Edit**.

Для запуска Registry Editor (Редактора реестра) и указания в нем компонентов:

1. В меню кнопки **Start** выберите пункт **Run**, а затем введите `RegEdit` в диалоговом окне **Run**.
2. В Registry Editor нажмите клавишу <F3> для открытия диалогового окна **Find**.
3. В поле **Find what** введите `CIPMReceive.CIPMRcv`. Сбросьте флажки **Keys** и **Value** (осуществляется только поиск данных), а затем щелкайте мышью по кнопке **Find Next**, пока не найдете вхождение с идентификатором класса (clsid, Class ID) для данного компонента.
4. В левом окне щелкните правой кнопкой мыши на элементе **clsid** вашего компонента, а затем выберите команду **Copy Key Name** в контекстном меню.
5. Закройте Registry Editor.

Для модификации значений в файле Affinity.reg и обновления реестра:

1. В файле Affinity.reg выделите часть первой строки: HKEY_CLASSES_ROOT\CLSID\{xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx}, в меню **Edit** выберите команду **Paste** для вставки **clsid** вашего компонента в файл.
2. Повторите эту процедуру и вставьте этот же **clsid** во вторую строку. Идентификатор категории (category ID) ступени включения в состав приложения (application integration stage) введен.
3. Сохраните файл и закройте программу Notepad.
4. В Microsoft Windows NT Explorer щелкните правой кнопкой мыши на файле Affinity.reg, а затем выберите команду **Merge** для обновления реестра.

Примечание

Следующая процедура должна выполняться только на компьютере-приемнике. Хотя ее нельзя осуществить на одном компьютере, данный компонент можно скомпоновать, инсталлировать, а затем включить в конвейер.

Для проверки компонента:

1. Откройте меню кнопки **Start**, укажите пункт **Programs**, потом последовательно команды **Microsoft Site Server** и **Commerce**, а затем щелкните **CIP Manager**.
2. В управляющем дереве последовательно раскройте **CIP Manager**, **CIP Manager (local)** и **Ramona**, выделите папку Document Profiles, щелкните правой кнопкой мыши на элементе **incoming document** (свойства входящих документов), а затем выберите команду **Properties**.
3. На вкладке **General** компонента включения в состав приложения (application integration component) выберите **CIPMReceive.CIPMRcv**, а затем щелкните **Properties**.
4. На странице свойств введите `C:\TestApp.txt` в соответствующем поле, а затем нажмите кнопку **OK**. Снова нажмите кнопку **OK** для подтверждения установок.

Для просмотра результата на передающем компьютере:

1. Перейдите в папку Labs\Lab05\TransmitApplication и дважды щелкните на файле CIPMTransmit.exe.
2. Заполните поля формы, включите книги в покупательскую корзину, а затем щелкните по **Place Order** (Поместить заказ) для отправки детальной информации о покупке на компьютер-приемник. Вы должны увидеть окно с сообщением: "The book order has been sent" (Заказ на книгу отправлен).

Для просмотра результата на компьютере-приемнике в Microsoft Windows NT Explorer под корневой папкой диска C: дважды щелкните мышью на

файле TestApp.txt для просмотра данных, отправленных вам с передающего компьютера.

Вопросы для повторения

1. Перечислите те элементы, для настройки и работы которых с программой CIP Manager необходимо использовать Visual Basic.
2. Какая информация содержится в наборе коммерческих параметров (Trading Profile)?
3. Какие методы связаны с объектами `SendReceive`?
4. Какую ступень указывает идентификатор категории (category ID) в reg-файле?

ГЛАВА 6

Обзор XML

Краткий обзор

XML (eXtensible Markup Language, язык расширяемой разметки) разработан с целью облегчения разработки Web-сайтов, расширения их продуктивности и функциональных возможностей. В этой главе вы познакомитесь с XML, *определением типа документа* XML (DTD, Document Type Definition), и с типами данных, с которыми он может работать.

Примечание

В данной главе в качестве XML-процессора используется Microsoft Internet Explorer. Он соответствует требованиям рекомендаций консорциума W3C от 10 февраля 1998 г. XML-схемы (XML Schemas) соответствуют требованиям, изложенным в рабочих материалах консорциума W3C (W3C XML Schema Working Drafts) от 6 мая 1999 года.

Цели и задачи

После прочтения данной главы вы научитесь:

- создавать XML-документ;
- создавать DTD XML;
- осуществлять проверку действительности (validate) XML-документа с помощью DTD;
- представлять наиболее распространенные типы данных в XML.

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod06\Mod06Code.txt.

Язык XML

XML представляет собой упрощенный вариант SGML (Standard Generalized Markup Language, стандартного обобщенного языка разметки), позволяющий разработчикам и администраторам Web-сайтов методично структурировать данные, размещенные на Web-странице.

Переход от HTML к XML

Язык разметки гипертекста (HTML, Hypertext Markup Language) широко используется разработчиками и распознается большинством браузеров. Хотя HTML предоставляет возможность управления отображением данных на экране, он не позволяет создавать теги. Это вызывает определенные проблемы у разработчиков при поиске и извлечении данных в Web.

На рис. 6.1 представлено HTML-отображение XML-документа, рассматриваемого в данном разделе.

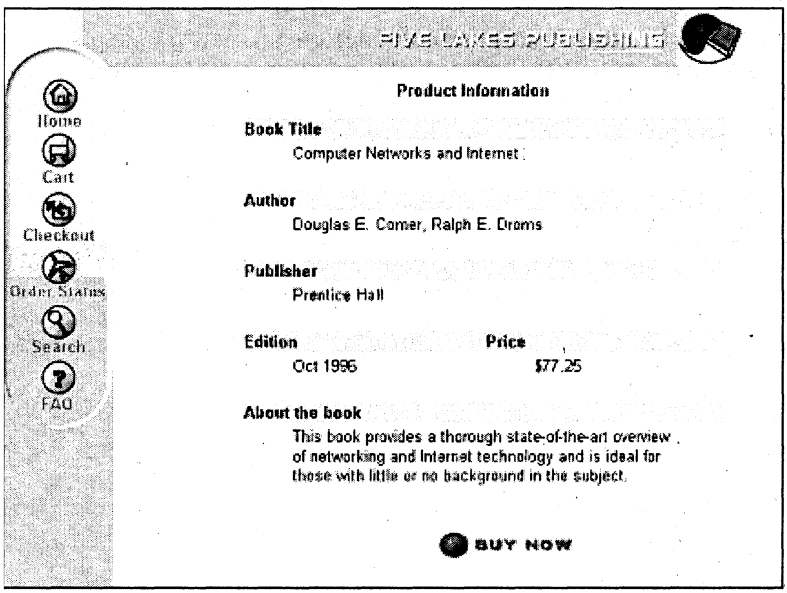


Рис. 6.1. HTML-отображение XML-документа

Разберем фрагмент HTML-кода для Web-страницы, на которой представлена часть типового библиографического описания книги с указанием ее издателя.

```
<tr>
  <td align="left" bgColor="#D7FFD7" width="391" colspan="2">
```

```
    <font face="Arial" size="2">
      <strong>Publisher </strong>
         
    </font>
  </td>
</tr>
<tr>
  <td align="left" bgColor="#FFFFD0" width="391" colspan="2">
    <blockquote>
      <p>
        <font face="arial" size="2">Prentice Hall</font>
      </p>
    </blockquote>
  </td>
</tr>
```

Представленная разметка не позволяет создать программу, осуществляющую поиск наименования издательства. Поэтому разработчику необходимо создать программу, которая будет удалять теги и возвращать лишь наименование издательства, в данном случае — Prentice Hall. Ввиду того, что разные Web-страницы, содержащие библиографические описания книг, могут иметь различную разметку, программа поиска должна быть очень гибкой.

Преимущества XML

HTML является примером простого подмножества SGML. XML мощнее, чем HTML, т. к. в нем в большей мере реализованы рекомендации SGML. Некоторые преимущества XML перед HTML приведены в табл. 6.1

Таблица 6.1. *Преимущества XML*

HTML	XML
Разметка описывает способ отображения данных в браузере	Разметка описывает сами данные
Разработчики вынуждены пользоваться ограниченным набором заранее определенных тегов	Разработчики создают теги и поэтому могут описывать любые типы данных
Поиск и обновление данных затруднено, т. к. один и тот же набор тегов может применяться для различных типов данных	Поиск и обновление данных осуществляется легко, поскольку для описания любого типа данных имеется свой уникальный набор тегов
В браузере производится управление способом отображения данных	XML не осуществляет управления способом отображения данных

Отображение XML-кода в браузере

Хотя XML предоставляет разработчикам возможность управления данными, он не позволяет управлять их отображением в браузере. Для представления данных XML-документ сначала должен быть связан с таблицей стилей. "Родным" языком таблиц стилей в XML является так называемый *расширяемый язык стиля* (XSL, Extensible Style Language). Кроме того, в XML предусмотрена поддержка *каскадных таблиц стилей*[1] (CSS, Cascading Style Sheets).

Требования к XML-документам

Любой объект данных, представляющий собой XML-документ, должен отвечать двум основным требованиям. Во-первых, он должен быть *правильным* (well-formed). Во-вторых, объект данных может быть также *действительным* или *допустимым* (valid). Для формирования XML-документа необходимо, чтобы объект данных удовлетворял ограничению, известному в терминологии XML как *ограничение по правильности* (well-formedness constraint). Кроме того, документ может также удовлетворять *ограничению по действительности (допустимости)* (validity constraint). (Недействительный (invalid) документ не воспринимается XML-процессором как фатальная ошибка.) Следующий фрагмент кода отображает базовую структуру XML-документа:

```
<? xml version="1.0" standalone="yes" ?>
  <root_tag>
    <element_1>
      This is a sample XML Document
    </element_1>
    <message text="Hello World" />
  </root_tag>
```

Правильность XML-документа

Для того чтобы объект данных можно было считать правильным XML-документом, он должен отвечать следующим основным критериям:

- документ должен начинаться с объявления XML `<?xml version="1.0"?>`;
- документ должен иметь единый корневой элемент (root element);
- все другие элементы должны находиться внутри этого корневого элемента и его закрывающего элемента (closing element);
- элементы документа должны быть правильно вложены (nested).

Для того чтобы соответствовать критерию правильности (well-formedness), в нем должны выполняться правила, относящиеся к структуре документа,

---

[1] Для создания таблиц стилей XML-документов применяется также язык описания и семантики стиля документов (DSSSL, Document Style Semantics and Specification Language). — *Пер.*

чувствительности регистра в XML, описанию значений атрибутов и управлению пробельными символами.

- Все XML-документы являются структурированными. Это означает, что для процессора необходимо, чтобы каждому открывающему тегу (start tag) соответствовал закрывающий тег (closing tag). Неправильная вложенность тегов и пропуск закрывающих тегов должны вызывать сообщения XML-процессора о фатальной ошибке. Например, следующая разметка вызовет сообщение о фатальной ошибке:

```
<book_description>
  <author> Ralph E. Droms
</book_descripton>
  </author>
```

Примечание

Тег `<cover_image url="cover.gif" />` является пустым тегом. XML требует, чтобы последним символом в пустом теге (перед закрывающей угловой скобкой) обязательно был символ /. Этот символ указывает процессору на то, что данный тег является пустым, тем самым устраняя необходимость в отдельном закрывающем теге.

- XML является чувствительным к регистру. Таким образом, теги `<book_description>` и `</Book_description>` являются открывающим и закрывающим тегами двух разных элементов. Следующая разметка вызовет сообщение XML-процессора о фатальной ошибке:

```
<book_description>
  <author>Ralph E. Droms</author>
</Book_descripton>
```

- Обратите внимание, что в пустом теге `<cover_image url="cover.gif" />`, значение `"cover.gif"` атрибута `url` заключено в кавычки. Это является обязательным требованием к XML-документам. Если значение атрибута не будет заключено в кавычки, то XML-процессор выдаст сообщение о фатальной ошибке.

- В XML-документе разметка служит для определения данных. В отличие от HTML, где пробельные символы игнорируются, XML поддерживает пробельные символы. Таким образом, в следующих вариантах разметки представлены разные данные:

```
<book_title>Computer Networks and Internet</book_title>

<book_title>Computer Networks and
    Internet</book_title>
```

Допустимость XML-документа

Так как разработчики Web обычно имеют дело с сотнями страниц в Web-приложении, то очень легко забыть ключевые данные в любой из них. Например, на странице описания книги данные, содержащиеся в теге `<book_title>`, могут быть по невнимательности пропущены. Ограничения по действительности (допустимости) предназначены для проверки пропуска данных и сообщения разработчику о факте обнаружения такой ситуации. Это осуществляется путем указания в DTD данных, которые должны содержаться на странице.

Проверка допустимости XML-документов в Microsoft Internet Explorer 5.0

В окончательной версии Internet Explorer 5.0 для управления обработкой DTD используются два свойства.

- `ValidateOnParse`. Определяет, будут ли поступать пользователю сообщения об ошибках при проверке на допустимость.
- `ResolveExternals`. Определяет, загружено ли DTD или схема XML (XML Schema), и соблюдаются ли типы данных, значения, принятые по умолчанию и т. д.

В Internet Explorer 5.0 значением по умолчанию свойства `Validateonparse` является `false`. Пользователи должны устанавливать значение этого свойства в `true` с помощью языка скриптов, например, Microsoft JScript.

Логические структуры и теги

В следующем разделе рассматриваются логические структуры и теги, используемые в XML по умолчанию.

Логические структуры

Каждый XML-документ состоит из одного и более элементов. Эти элементы размещены между начальным и оконечным тегами или же в тегах пустого элемента. Логическая структура включает в себя элементы, *объявления типа элементов* (element type declarations), а также *объявления списка атрибутов* (attribute list declarations).

Открывающий, закрывающий теги и тег пустого элемента

Каждый элемент в XML-документе начинается с так называемого открывающего тега (start tag) и завершается закрывающим тегом (end tag). Закрывающий тег начинается с наклонной черты / и имеет то же самое имя и регистр, что и открывающий тег. Текст, размещенный между этими двумя

тегами, называется *содержимым элемента* (element's content). Элемент, который не имеет содержимого, называется *пустым элементом* (empty element).

Например, рассмотрим следующий код:

```
<book_description>
  <cover_image url="cover.gif" />
</book_description>
```

Эти теги и их ограничения представлены в табл. 6.2.

Таблица 6.2. *Ограничения на использование тегов в XML*

Ограничение	Пояснение

Уникальный спецификатор атрибута (unique attribute spec)	Имя атрибута не может появляться более одного раза в одном и том же открывающем теге или в теге пустого элемента. Следующая разметка приведет к фатальной ошибке, т. к. атрибут `name` повторяется в теге `book_name`:
	`<book_name name="..." author="..."_`
	`name="...">`
	`...`
	`</book_name>`

Не допускаются ссылки на внешние сущности[1] (external entity references)	Значения атрибута не могут содержать прямые или косвенные ссылки на сущности, объявленные вне активного документа или его DTD. Поэтому, если только сущность `&mr;` не была объявлена в активном документе или его DTD, следующая разметка должна вызвать со стороны XML-процессора сообщение о фатальной ошибке:
	`<book name="..." publisher=&mr;>`
	`...`
	`</book>`

Не допускается использование левой угловой скобки < в значениях атрибута	Замещающий текст любой сущности, на которую имеется прямая или же косвенная ссылка в значении атрибута, не должна содержать символа <. Единственным исключением является замещающий текст сущности `<`
Тип значения атрибута	Атрибут должен быть объявлен, и его значение должно соответствовать типу, объявленному для этого атрибута

[1] На самом деле, перевод "сущности" не отражает смысла этого термина. Лучше было бы сказать "подстановочный объект". Но, поскольку в литературе встречается именно такой перевод, будем использовать его и в этой книге. — *Ред.*

Первые три ограничения, представленные в предыдущей таблице, относятся к ограничениям правильности (well-formedness constraints). XML-процессор выдаст сообщение о фатальной ошибке, если будет нарушено любое из этих ограничений. Четвертое ограничение является ограничением допустимости (validity constraint), и процессор, если для него не будет соответствующих указаний, может игнорировать нарушения этого ограничения.

Теги, используемые по умолчанию

Хотя XML позволяет использовать теги для определения данных, существует несколько предопределенных конструкций. К ним относятся комментарии, *инструкции по выполнению* (PIs, processing instructions), а также разметка идентификации языка (language identification markup).

- XML-процессор игнорирует теги комментариев, так же как в HTML. Объявление комментариев в XML аналогично объявлению комментариев в HTML:

```
<!— This is a comment line —>
```

- Инструкции по выполнению являются инструкциями для приложений. Эти инструкции содержатся в разметке XML. Например, следующая разметка указывает, что данный документ соответствует спецификации XML 1.0 консорциума W3C:

```
<?xml version="1.0" standalone="yes"?>
```

Примечание

Выражение `standalone="yes"` указывает, что для проверки допустимости XML-документа и его модификации другие документы не используются. Если для проверки допустимости применяется DTD, или же для преобразований используется таблица стилей, то выражение `standalone` должно иметь следующий вид: `standalone="no"`.

- Идентификация языка (language identification) в XML задается при помощи атрибута `xml:language`. Этот необязательный атрибут принимает код, идентифицирующий язык кодирования данных.

  Например, для указания, что абзац данных представлен на английском языке, необходимо использовать следующую разметку:

```
<p xml:language="en"> ... </p>
```

Полные справочные данные по разрешенным кодам языков содержатся в документации стандарта ISO 639 "Codes for the representation of names of languages" (Коды, используемые для представления наименований языков).

Анализ кода XML-документа

Далее вы сможете изучить основную структуру XML-документа.

HTML-страница

На рис. 6.1 представлено HTML-отображение XML-документа в коде, приведенном ниже.

Пример XML-документа

Страница описания книги в HTML-коде может быть представлена в XML следующим образом[1]:

```
<?xml version="1.0" standalone="yes" ?>
<!- book_description - описание книги ->
  <book_description>

     <!- book_title - наименование книги ->
     <book_title>Computer Networks and Internet</book_title>

     <!- author - автор ->
     <author>Douglas E. Comer</author>
     <author>Ralph E. Droms</author>

     <!- publisher - издательство ->
     <publisher>Prentice Hall</publisher>

     <!- edition - дата издания ->
     <edition>October 1996</edition>

     <!- price - стоимость ->
     <price>77.25</price>

     <!- cover_image url - URL изображения, представленного на обложке ->
     <cover_image url="cover.gif"/>

     <!- review written_by  - "Кем составлена аннотация" ->
     <review written_by="Publisher">

          <!- Текст аннотации: ->
          This book provides a thorough state-of-the-art overview
          of networking and Internet technology and is ideal for
          those with little or no background in the subject.

     </review>
  </book_description>
```

[1] Перед каждой строкой листинга, там, где необходимо, приводится комментарий, содержащий перевод наименований тегов и содержания элементов. — *Пер*

Обратите внимание на следующие требования, встретившиеся в данном примере.

- Первым объявлением в документе является инструкция XML-процессору:

  ```
  <?xml version="1.0" standalone="yes" ?>
  ```

  Она содержит указание XML-процессору о том, что данный документ отвечает требованиям спецификации XML 1.0 и к нему не присоединены какие-либо другие документы.

- Открывающим тегом корневого элемента документа является `<book_description>`, а все остальные теги вложены в корневой элемент.

- Каждому открывающему тегу соответствует закрывающий тег. Единственным исключением является тег `<cover_image url="cover.gif" />`. Поскольку он представляет собой пустой тег и в его объявлении используется наклонная черта /, то такой тег допустим, и для него закрывающий тег не нужен.

- Во всех элементах, ограниченных парами тегов, содержатся данные.

Данный XML-документ является правильным и допустимым, т. к. в нем соблюдаются все указанные выше ограничения.

Описание типа документа

После создания XML-документа можно осуществить проверку его допустимости. Для этой цели используется DTD. В DTD содержатся объявления всех элементов XML-документа, правила, задающие частоту и необходимость их присутствия, а также допустимые значения атрибутов. Далее вы научитесь объявлять DTD и правила для элементов.

Физические структуры в XML

Помимо логических структур, рассмотренных ранее в этой главе, в XML имеется ряд физических структур (physical structures). К ним относятся тип элемента (element type), список атрибутов (attribute list) и объявления сущностей (entity declarations)

Объявления типа элементов и списка атрибутов

Объявления типа элементов и списка атрибутов, содержащиеся в DTD, используются для проверки допустимости XML-документов.

Объявление типа элемента задает ограничения на содержание элемента. Объявление списка атрибутов определяет набор атрибутов для заданного

типа элемента и накладывает ограничения, а также значения по умолчанию для этих атрибутов. Далее в настоящей главе вы более подробно изучите эти объявления. В следующем фрагменте разметки представлены примеры объявлений типа элемента и списка атрибутов:

```
<!ELEMENT br EMPTY>
<!ATTLIST list
    type "ordered">
```

Объявление сущностей

Физическая структура XML-документа состоит из одной и более так называемых *сущностей* (entities). Эти сущности могут как обрабатываться, так и не обрабатываться[1] синтаксическим анализатором (parser) и быть объявлены как внутренние (internal), либо как внешние (external) сущности.

Внутренние сущности

Внутренние сущности (internal entities) всегда обрабатываются анализатором. У них нет отдельных объектов для хранения значений. Содержимое этих значений расположено непосредственно в самом объявлении сущности, как показано в следующем примере:

```
<!ENTITY Edition "This is a First Edition Collectors issue">
```

Внешние сущности

Все сущности, которые не являются внутренними, относятся к *внешним сущностям* (external entities). У них есть ограничение значения, которое задает соответствие между именем (name) и объявленным именем условного обозначения, как показано в следующем примере:

```
<!ENTITY cover-pic SYSTEM "../images/cover.gif">
```

Неанализируемые сущности

Содержимое неанализируемых сущностей может быть текст. Если тип содержимого — текст (text), то это содержимое может, но не обязательно должно представлять собой XML-код. XML-процессор лишь создает идентификаторы (identifiers) сущности и условное обозначение (notation), доступное приложению. Для сущностей, не обрабатываемых анализатором (неанализируемых сущностей), ограничения не задаются.

---

[1] В первом случае назовем их анализируемыми, а во втором — неанализируемыми сущностями. — *Пер.*

Анализируемые сущности

Содержимое анализируемой сущности называется *замещающим текстом сущности* (entity's replacement text). Этот текст является неотъемлемой частью XML-документа. Анализируемые сущности должны быть правильными (well-formed). В результате, логические и физические структуры в XML-документе являются правильно вложенными.

```
<!ENTITY file1 SYSTEM "file1.xml">
```

В каждой внешней анализируемой сущности может применяться различная кодировка символов. Поэтому XML-процессоры должны распознавать сущности, представленные в форматах UTF-8 или UTF-16. Анализируемые сущности, представленные в других форматах кодирования, должны начинаться с объявления системы кодирования (encoding declaration). Ниже приводится пример такого объявления:

```
<?xml encoding='EUC-JP'?>
```

Анализ кода для DTD

DTD может объявляться как часть самого XML-документа, или же как отдельный документ, на который ссылаются различные XML-документы[1]. Формат объявления внутреннего DTD был представлен выше. Однако объявление внешнего DTD будет выглядеть не так, как в данном примере:

```
<!DOCTYPE book [

...
]>
```

Напротив, вызов внешнего DTD в XML-документе будет иметь следующий вид:

```
<? xml version="1.0" standalone="no" ?>
<!DOCTYPE book_description SYSTEM "book.dtd">
```

Пример внешнего DTD

Приведем пример внешнего DTD для документа, содержащего описание книги:

```
<!ELEMENT book_description (book_title,_
    author+, publisher, edition?,_
    price, cover_image?, review*)>
<!ELEMENT book_title (#PCDATA)>
```

---

[1] В первом случае DTD называют внутренними (internal DTD), а во втором — внешними (external DTD). — *Пер.*

```
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT cover_image (#PCDATA)>
<!ELEMENT review (#PCDATA)>
```

В представленном выше DTD, обратите внимание на следующее объявление[1]:

```
<!ELEMENT book_description (book_title, ↳
              author+, publisher, edition?, ↳
              price, cover?, review*)>
```

Это объявление указывает, что в XML-документе, ссылающемся на данное DTD, имеется корневой элемент `book_description`. Оно также указывает, что допустимыми дочерними элементами являются элементы `book_title`, `author`, `publisher`, `edition`, `price`, `cover`, `review`.

В этой же разметке также заданы правила проверки действительности дочерних элементов, с помощью символов, значение которых представлено в табл. 6.3.

Таблица 6.3. *Значения символов*

Символ	Значение
+	Если за именем элемента следует данный символ, то это означает, что данный элемент должен встречаться в XML-документе один и более раз
?	Если за именем элемента следует данный символ, то это означает, что данный элемент может встречаться в XML-документе один или ни одного раза
*	Если за именем элемента следует данный символ, то это означает, что элемент может не встречаться в XML-документе ни одного раза, встречаться один раз или же любое количество раз

В DTD, представленном в предыдущем примере, обратите внимание на следующее объявление:

```
<!ELEMENT book_title (#PCDATA)>
```

В этом объявлении сообщается, что элемент `book_title` не содержит каких-либо дочерних элементов. В нем также указывается, что данные в этом элементе могут включать в себя как текст, так и разметку. В том случае, если

[1] Здесь и далее символ ↳ означает, что код должен размещаться в той же строке, где находится указанный символ. — *Ред.*

бы данные включали в себя только текст, использовалось бы следующее объявление:

```
<!ELEMENT book_title (#CDATA)>
```

Ни в одном из элементов, объявленном в предыдущем DTD, не содержатся подэлементы (subitems). Поэтому указано, чтобы в этих элементах присутствовали анализируемые символьные данные (PCDATA). Например, если бы в документе, содержащем описание книги, у элемента `book` было два дочерних элемента — `ISBN` и `Title`, то в DTD для них использовалось бы следующее объявление:

```
<!ELEMENT book_description (book, ...)>
  <!ELEMENT book (ISBN, Title)>
    <!ELEMENT ISBN (#PCDATA)>
    <!ELEMENT Title (#PCDATA)>
```

В следующем листинге представлено все, что обсуждалось в данном разделе:

```
<!DOCTYPE book[
  <!ELEMENT book_description (book_title,
            author+, publisher, edition?,
            price, cover_image?, review*)>
<!ELEMENT book_title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>

<!ELEMENT edition (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT cover_image (#PCDATA)>
<!ELEMENT review (#PCDATA)>
]>
```

Объявления списка атрибутов и сущностей в DTD

DTD в XML не только определяет элементы, которые допускается применять в каком-либо XML-документе и заданные для них ограничения, но и наращивает мощь и полезность XML, благодаря использованию объявлений списка атрибутов (attribute list) и сущностей.

Объявление списка атрибутов в DTD

Объявления списка атрибутов эффективно используются для усиления возможностей выборки данных, а также для проверки, что документ является допустимым (valid). Ниже дан пример разметки аннотации к книге:

```
<review>
    By Publisher: This book provides a thorough state-of-the-art
```

```
    overview of networking and Internet technology and is ideal
    for those with little or no background in the subject.
</review>
```

Представленная выше разметка не предоставляет возможности управления данными об авторе аннотации. Для доступа к этим данным может использоваться следующий вариант разметки:

```
<review written_by="publisher">
    This book provides a thorough state-of-the-art overview of
    networking and Internet technology and is ideal for those
    with little or no background in the subject.
</review>
```

Атрибут `written_by` будет полезным лишь в том случае, если приложение произведет считывание значения этого атрибута.

Прежде чем атрибут будет включен в допустимый XML-документ как его часть, его необходимо вначале объявить в DTD, используя для этого объявление списка атрибутов (attribute list declaration). Например, в DTD, представленном ранее в данном разделе, можно было бы воспользоваться следующим объявлением:

```
<!ELEMENT review #PCDATA>
  <!ATTLIST review written_by CDATA #REQUIRED>
```

В предыдущем объявлении указано, что элемент `review` имеет атрибут `written_by`, который принимает символьные данные (CDATA, character data) и является обязательным атрибутом. В табл. 6.4 содержится список ключевых слов, используемых для объявления атрибутов.

Таблица 6.4. *Список ключевых слов*

Ключевое слово	Значение
`#REQUIRED`	Для атрибута элемента должно быть задано значение
`#FIXED`	Атрибут имеет значение, которое не меняется в пределах документа
`#IMPLIED`	Атрибут может иметь объявленное значение, или же оставаться пустым
Значение по умолчанию (default value)	Это значение указано вместо ключевого слова в теге `<!ATTLIST>`. Если некоторое значение задано в документе, то оно замещает значение, принятое по умолчанию. Однако если в документе значение атрибута не задано (оставлено пустым), то предполагается значение, принятое по умолчанию

Для редактирования существующих DTD вам необходимо знать четыре основных типа атрибутов. Они представлены в табл. 6.5.

Таблица 6.5. *Основные типы атрибутов*

Тип атрибута	Описание
CDATA	Указывает XML-процессору, что в качестве значения атрибута будут использоваться текстовые данные
ID	Задает уникальный идентификатор для атрибута. Имена ID должны начинаться с буквы или же с символа подчеркивания, и не могут содержать пробелов
NMTOKEN	Указывает на то, что атрибут должен начинаться с буквы или символа подчеркивания, и при этом не может содержать пробелов. Однако здесь допускается дублирование значений
Enumerated (перечислимый тип)	Этот тип не имеет ключевого слова. Он состоит из списка всех возможных значений, помещённых внутри круглых скобок и разделенных символом |, как в следующем примере: `<!ATTLIST Binding (soft | hard) #REQUIRED>`

Примечание

Более детальная информация о DTD XML представлена в документах консорциума W3C на Web-сайте **http://www.w3.org/**.

Сущности в DTD

Определение сущностей

В XML имеется пять предопределенных сущностей (табл. 6.6).

Таблица 6.6. *Предопределенные сущности*

Символ	Объявление сущности
&	`&`
<	`<`
>	`>`
"	`&quot;`
'	`&apos;`

XML также предоставляет возможность задавать и другие сущности в DTD. Например, если слова `"Five Lakes"` часто используются на сайте, то они могут быть представлены с помощью сущности `"&fl;"`. Для этого используется следующее объявление:

```
<!ENTITY fl "Five Lakes">
```

Сохранение данных в файлах

Большие строки данных могут храниться в отдельном файле и объявляться путем указания ключевого слова XML — "SYSTEM", которое располагается между именем сущности файла и URL, как показано в следующем примере:

```
<!ENTITY text SYSTEM "http://list.txt">
```

Раскрытие параметрической сущности

Если объявление сущности содержит субвхождения (subentries) или опции (options), то такую сущность нельзя "раскрыть" в DTD. Для того чтобы это сделать, вам необходимо объявить эту сущность при помощи символа %, который нужно вставить непосредственно перед именем сущности. Чтобы можно было раскрыть такую сущность (она называется *параметрической* (parameter entity)), ее имени должен предшествовать символ процента, а в конце имени должна располагаться точка с запятой. Ниже показано использование параметрических сущностей на примере описания визитной карточки:

```
<!ELEMENT visiting_card (%property.man;, (%property.personal;
| property.business;)
```

Когда вы определяете корневой элемент, информация визитной карточки хранится в различных параметрических сущностях. Для определения сущности property.personal применяется следующая разметка:

```
<!ENTITY %property.personal "(nickname | photograph | birthdate)>
```

Использование параметрических сущностей облегчает чтение объявлений элементов.

XML-данные

XML-документ представляет собой правильный (well-formed) объект данных, что позволяет легко представлять в XML различные типы данных. В этом разделе вы узнаете, как в XML представлены некоторые распространенные типы данных. Один из таких типов данных именуют *схемой XML-данных* (XML-Data Schema). Этот тип данных может использоваться для определения XML-документов и, следовательно, может применяться в качестве замещения в DTD. Предлагаемый далее материал посвящен подробному рассмотрению указанного типа данных.

Представление данных в XML

В XML могут быть легко представлены следующие распространенные типы данных:

- данные (Data);
- метаданные (Metadata);
- цифровые подписи (Digital signatures);
- информация базы данных (Database information);
- графические структуры (Graph structures);
- разобщенная информация (Discontiguous information (propertyOf));
- схема (Schema);
- расширение типа (Type extension).

Далее будет рассмотрена реализация лишь трех типов данных.

Данные

XML-данные могут быть неоднородными по своей структуре. Например, книги, музыкальные записи и кофе могут быть закодированы в заказе на покупку следующим образом[1]:

```
<!- order - заказ;
    customer - покупатель;
    last_name - фамилия;
    first_name - имя;
    item - номенклатура товара;
    type - тип;
    book - книга;
    title - название;
    price - цена;
    record - музыкальная запись;
    media - среда (носитель записи);
    album - альбом;
    artist - исполнитель;
    cups - чашки ->
<order>
  <customer>
    <last_name>Doe</last_name>
    <first_name>John</first_name>
  </customer>
    <item type="book">
      <title>Computer Networks and Internet</title>
      <price>77.25</price>
    </item>
```

[1] Чтобы не затруднять восприятие структуры XML-данных многочисленными комментариями, приведем перевод некоторых тегов в начале данного примера — *Пер.*

```
  <item type="record">
    <media>Compact Disc</media>
    <album>I could have coded all day</album>
    <artist>The debuggers</artist>
    <price>25.75</price>
  </item>
  <item type="coffee">
    <type>Decaff</type>
    <cups>1</cups>
  </item>
</order>
```

Метаданные

XML может представлять сложные, независимые структуры данных. Он также позволяет описывать информацию об удаленных Web-ресурсах, многие из которых сами представляют собой некоторые данные. Рассмотрим следующий пример:

```
<e-partner>
  <item href="http://www.abc-press.com/welcome.htm">
    <a href="http://www.abc-press.com/instock.htm">
       Check e-partner for available stock
    </a>
    <title>Welcome to the ABC Press site</title>
  </item>
</e-partner>
```

В этом фрагменте XML-документа, для получения необходимых данных, имеется ссылка на Web-сайт электронного партнера.

Информация базы данных

XML может представлять как сложные, так и простые структуры данных. Например, для представления списка записей базы данных может использоваться следующая разметка:

```
<out_of_stock>
  <item type="book">
    <name>Programming Visual C++,</name>
    <vendor>Microsoft Press</vendor>
  </item>
  <item type="records">
    <name>Wounded Knees and Broken Elbows</name>
    <vendor>The Garage CD-dump</vendor>
  </item>
</out_of_stock>
```

Преобразование данных в XML и обратное их извлечение из структур XML-данных может осуществляться с помощью технологии активных серверных страниц (ASP, Active Server Pages). Источник данных, созданный с помощью ASP, будет оставаться ASP-страницей. Однако данная ASP-страница будет осуществлять вывод доступного XML-кода на сервере.

Имеется два преимущества при создании источника XML-данных с помощью ASP-страницы для обращения к базе данных. Во-первых, скрипт может осуществлять конвертирование больших объемов по мере того, как он станет производить заполнение источника данных. Во-вторых, источник данных будет автоматически обновляться по мере обновления базы данных.

Ограничения DTD

Главный недостаток DTD заключается в том, что они не основываются на XML. Следовательно, для проверки действительности XML-документов с помощью DTD, программисты должны знать, как писать действительные правила в DTD. К тому же, DTD не предоставляет программистам возможности контроля за типами данных, которые могут принять элементы. Схема XML-данных преодолевает эти недостатки DTD.

В данном разделе представлен пример схемы XML-данных. Вы сможете более подробно изучить объявления схемы данных в более углубленном курсе.

```
<xml:schema ID="ArtSchema">
  <elementType id="artistic-work">
    <relation href="#title" />
  </elementType>
<elementType id="book" extends="#artistic-work">↳
  <relation href="author" />
</elementType>

<relationType id="author">
    <pcdata />
</relationType>
```

Анализ схемы

В представленном выше примере обратите внимание на следующее:

- объявление `<elementType id="book" extends="#artistic-work">` указывает, что книги (`books`) являются подмножеством художественных работ (`artistic-works`);
- объявление `<relation href="author">` указывает, что элемент `book` имеет подэлемент `author`, входными данными для которого являются анализируемые символьные данные (PCDATA).

Дополнительные сведения

Для дальнейшего чтения рекомендуется обратиться к следующим URL:

- **http://msdn.microsoft.com/xml/default.asp**
- **http://www.xml.com/pub**
- **http://xmlephant.com/**
- **http://www.oasis-open.org/**
- **http://vbxml.com/**
- **http://xmlinfo.com/**

Лабораторная работа 6. Создание и проверка допустимости XML-документа, содержащего описание книги

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать XML-документ, содержащий описание книги;
- писать DTD для проверки действительности созданного вами XML-документа;
- связывать XML-документ и DTD.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- знать HTML;
- хорошо знать Microsoft Internet Explorer 5.0 final release или более поздние версии.

Сценарий

Web-сайт компании "Ramona Publishing" преобразуется из HTML в XML. Вам дана страница этого сайта, содержащая описание книги. Сначала вы соберете данные на этой странице и преобразуете их в XML-документ. Далее, вы составите DTD, которое может использоваться для проверки допустимости данного XML-документа. Наконец, вы сможете связать это DTD и XML-документ.

Примечание

В данной лабораторной работе в качестве символа продолжения строки используется символ ↳.

Примечание

Время, необходимое для выполнения данной работы, — 15 минут.

Упражнение 1. Создание XML-документа

В этом упражнении вы откроете HTML-страницу, содержащую описание книги, и скопируете находящиеся в ней данные в XML-документ.

Для того чтобы открыть HTML-страницу, содержащую описание книги, необходимо в Microsoft Windows NT Explorer открыть папку Labs\Lab06 \StartCode, а затем дважды щелкнуть по файлу book.htm. Данная страница будет отображена в Microsoft Internet Explorer.

Для того чтобы открыть XML-документ:

1. В меню кнопки **Start** укажите пункт **Programs**, выберите команду **Accessories**, а затем щелкните на **Notepad**.
2. В папке Labs\Lab06\StartCode откройте файл book.xml.

Для того чтобы завершить работу над XML-документом:

1. В файле book.htm найдите соответствующие значения, которые необходимо указать в файле book.xml.
2. Скопируйте каждое значение и вставьте их в файл book.xml.
3. Добавьте соответствующие закрывающие теги в XML-документ.
4. Сохраните файл book.xml в папке Labs\Lab06.

Для просмотра результатов в Windows NT Explorer откройте файл book.xml в папке Labs\Lab06. В браузере отображается XML-документ.

Упражнение 2. Создание DTD XML

В этом упражнении вы сформируете DTD для XML-документа, который вы создали в предыдущем упражнении.

Для того чтобы сформировать DTD:

1. Создайте новый файл в программе Notepad и добавьте следующие объявления:

```
<!ELEMENT book_description (Book_Title, Author+, ↳
Edition?, Price, Cover_image?, About_the_Book*)>
  <!ELEMENT Book_Title (#PCDATA)>
```

```
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Edition (#PCDATA)>
<!ELEMENT Price (#PCDATA)>
<!ELEMENT Cover_image (#PCDATA)>
<!ELEMENT About_the_Book (#PCDATA)>
```

2. Сохраните данный файл под именем book.dtd в папке Labs\Lab06.

Упражнение 3. Связывание XML-документа и DTD

В этом упражнении вы свяжете созданный вами XML-документ с тем DTD, который вы создали в предыдущем упражнении.

Для того чтобы открыть XML-документ, в программе Notepad откройте файл book.xml из папки Labs\Lab06.

Для того чтобы связать данный документ с DTD:

1. Найдите следующую разметку в документе:

```
<?xml version="1.0" ?>
```

2. Сразу же за этой разметкой вставьте такой код:

```
<!DOCTYPE book_description SYSTEM "book.dtd">
```

3. Сохраните и закройте файл.

Для просмотра результата в Internet Explorer откройте файл Labs\Lab06 \book.xml. В браузере будет отображен XML-документ в виде сворачиваемого дерева данных.

Вопросы для повторения

1. Каким требованиям должен отвечать объект данных для того, чтобы быть XML-документом?
2. Как используется определение типа документа (DTD, Document Type Definition)?
3. В чем заключается разница между HTML и XML?
4. Какие ограничения DTD привели к разработке схем XML-данных (XML-Data Schemas)?

ГЛАВА 7

Обзор расширяемого языка стилей XSL

Краткий обзор

После создания и проверки допустимости XML-документа необходимо обеспечить отображение XML-данных. Хотя теги XML представляют структуру данных, они не могут управлять способом отображения последних. Для того чтобы отобразить данные, содержащиеся в XML-документе, XML-разметка должна быть преобразована в HTML-разметку. Это можно сделать с помощью языка таблиц стилей (style sheet language), например, такого как *каскадные таблицы стилей* (CSS, Cascading Style Sheets) или *языка семантики и описания стиля документа* (DSSSL, Document Style Semantics and Specification Language). Наиболее мощным языком таблиц стилей является *расширяемый язык стиля* (XSL, Extensible Stylesheet Language). Хотя XSL находится в стадии становления (formative stage), он специально создан для работы с XML.

Примечание

В этой главе в качестве XML-процессора используется Microsoft Internet Explorer. Он отвечает рекомендациям, изложенным в рабочих материалах по языку трансформации XSLT (XSL Transformations) консорциума W3C ("W3C Working Draft XSL Transformations (XSLT) Specification Version 1.0" от 21.04.99[1]) и в спецификации XSL ("Extensible Stylesheet Language(XSL) Specification" от 21.04.99)[2].

В этой главе вы познакомитесь с XSL и выясните, как он соотносится с CSS и DSSSL. Вы также узнаете об *объектах потока* (flow objects) *XSL* и *правилах*

---

[1] На момент выпуска книги существовали рабочие материалы "XSL Transformations (XSLT) Version 1.1" от 12.12.2000, найти которые можно по адресу **http://www.w3.org/TR/xslt11**. — *Ред.*

[2] Последние материалы датированы 12.11.2000. — *Ред.*

таблиц стилей (style sheet rules). Наконец, вы научитесь конвертировать XML-документы в HTML-страницы.

Цели и задачи

После прочтения данной главы вы научитесь:

- объяснять необходимость XSL;
- рассказывать о том, как формировался XSL;
- строить деревья результата с помощью XSL;
- создавать таблицы стилей XSL.

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod07\Mod07Code.txt.

XSL

Хотя XML позволяет структурировать данные, он не предоставляет вам возможности управлять отображением данных. XSL решает эту проблему — с его помощью можно указать, как именно будут отформатированы данные в XML-документе. Далее вы узнаете о преимуществах использования XSL и о том, как он развивается, позволяя преодолевать существующие ограничения языков CSS и DSSSL.

Общие сведения

XSL делает управление данными и их представление на Web-сайтах более простым и мощным, чем было ранее.

Анализ XSL

XSL состоит из двух частей. Первую часть представляет *язык трансформаций XSL* (XSLT, XSL Transformations), используемый для преобразования XML-документов. Вторую часть представляет словарь XML (XML vocabulary), предназначенный для описания семантики форматирования.

Преимущества использования XSL

Во время работы над данной книгой XSL по-прежнему находился в стадии разработки и дата выпуска его окончательной версии еще не обнародована. Когда, однако, эта версия будет доступной, XSL обеспечит следующие преимущества над другими языками скриптов.

- XSL специально создан с учетом требований, предъявляемых со стороны XML к языку определения стиля. Данные, обрабатываемые XSL-процессором, являются вычисленными (evaluated), переупорядоченными (rearranged) и перекомпонованными (reassembled). Благодаря этому создается гибкий источник информации, который легко поддается модификации и обновлению.
- XSL-документы являются допустимыми XML-документами. Таким образом, XSL представляется для Web-разработчиков в виде соответствующей разметки. Следовательно, любой, кто работал с подмножеством языков SGML, таких как, например, HTML, может легко освоить XSL.
- XSL позволяет исключить использование различных скрипт-языков, например, Perl, Microsoft Visual Basic Scripting Edition, Microsoft JScript, предоставляя тем самым практически неограниченную расширяемость (extensibility).

Эволюция XSL

XSL основывается на двух языках — CSS и DSSSL. Свыше 90% свойств XSL уже описано в CSS. XSL поддерживает все рекомендации CSS1.

Отличие XSL от CSS и DSSSL

Основным ограничением CSS является отсутствие поддержки разбиения на страницы (pagination). Для того чтобы предоставить возможность управления разбиением на страницы, в XSL был расширен набор объектов форматирования и свойств форматирования.

При разработке расширения CSS за основу был принят DSSSL. Однако реальное расширение не всегда выглядит как DSSSL. Для того чтобы добиться более точного соответствия спецификации CSS или же для упрощения работы в разных регистрах по сравнению с DSSSL, некоторые упрощения были унаследованы из CSS.

Изменения в каталоге CSS

В XSL было внесено два типа изменений по отношению к каталогам CSS: либо были модифицированы уже существующие свойства каталога, либо были созданы совершенно новые свойства. Эти изменения предоставляли в XSL возможность управления границами страниц, что отсутствовало в CSS, а также позволили управлять прокручиваемыми документами, что отсутствовало в DSSSL. Все это сделало XSL самым предпочтительным языком таблиц стилей для XML.

Понятия XSL

В процессе чтения XML-документа процессор формирует дерево, именуемое *исходным деревом* (source tree). Это дерево может иметь *узлы* (nodes), виды которых перечислены в табл. 7.1.

Таблица 7.1. *Типы узлов XML-документа*

Тип узла	Описание
Корень (root), корневой узел	Является корневым узлом дерева. Он не может встретиться еще где-либо в данном дереве. Корневой узел имеет лишь один дочерний узел для элемента такого типа документа, каким является XML-документ
Элемент (element), элементный узел	Этот тип узла создан для каждого элемента в документе. Его дочерними узлами могут быть узлы элементов, комментариев, информации по обработке, а также текстовые узлы, хранящие его содержимое
Текст (text), текстовый узел	Весь текст документа сгруппирован в виде текстовых узлов. Текстовому узлу не может непосредственно предшествовать, а также сразу же следовать за ним какой-либо другой текстовый узел
Атрибут (attribute), узел атрибута	Каждый элементный узел имеет множество узлов атрибутов. Значения атрибутов по умолчанию рассматриваются точно так же, как и указанные. У этих узлов нет дочерних узлов
Пространство имени (namespace), узел пространства имени	Для каждого элемента, начинающегося с последовательности `xlmns:`, а также для всех атрибутов такого узла, имеются узлы пространства имени. Эти узлы не имеют дочерних узлов
Инструкция по обработке (PI, Processing Instruction), узел инструкции по обработке	Для каждой инструкции по обработке имеется отдельный узел. Эти узлы не имеют дочерних узлов
Комментарий (comment), узел комментария	Для каждого комментария имеется узел комментария

На рис. 7.1 представлена структура таблицы стилей XSL.

Таблицы стилей XSL применяются по отношению к этому дереву для получения итогового дерева (result tree). В результате этого процесса узлы данного исходного дерева могут быть переупорядочены и продублированы.

Могут быть созданы новые элементы и объекты потока, такие как HTML-страница или, может быть, другое исходное дерево.

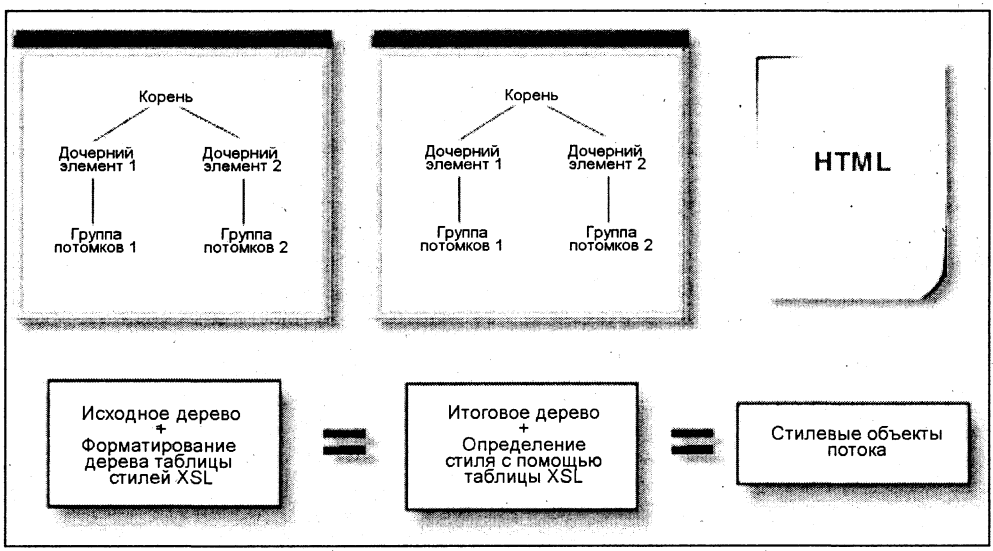


Рис. 7.1. Структура таблицы стилей XSL

Анализ кода таблицы стилей XSL

Для того чтобы отобразить в браузере XML-документ, содержащий описание книги, в этом документе необходимо использовать шаблон XSL. Ниже представлен пример такого шаблона:

```
<xsl:stylesheet
  xmlns:xsl="http://www.w3c.org/TR/WD-xsl"
  default-space "strip">
<xsl:template match="/">
  <HTML>
...
  <!— HTML-разметка -->
...
  <xsl:apply-templates select="book_description/book_title" />
  <xsl:apply-templates select="book_description/author" />
  <xsl:apply-templates select="book_description/publisher" />
  <xsl:apply-templates select="book_description/edition" />
  <xsl:apply-templates select="book_description/price" />
  <xsl:apply-templates select="book_description/review" />
...
```

```
  <!-HTML-разметка -->
  </HTML>
</xsl:template>
<xsl:template match="edition">
  <TD BGCOLOR="#FFFFD0" STYLE="width:391;text-align:left;
  font-family:Arial;font-size:10pt;" >
  <BLOCKQUOTE>
  <xsl:value-of />
  </BLOCKQUOTE>
  </TD>
</xsl:template>
<!-Другие шаблоны -->
</xsl:stylesheet>
  </TD>
</xsl:template>
<!-- Другие шаблоны ->
</xsl:stylesheet>
```

В представленной выше таблице стилей XSL, обратите внимание на следующие элементы:

- `xsl:stylesheet`
- `xsl:template`
- `xsl:value-of`

Элемент *xsl:stylesheet*

Элемент `xsl:stylesheet` является первым элементом, объявленным в данной таблице стилей. У него есть следующие атрибуты.

- `xmlns:xsl="http://www.w3c.org/TR/WD-xsl"` указывает процессору, что каждый элемент, который начинается с объявления `xsl:`, должен интерпретироваться в соответствии со спецификацией XSL. Другими словами, все элементы, начинающиеся с `xsl:`, принадлежат к пространству имен XSL (XSL Namespace).

Примечание

В качестве пространства имен, на которое ссылаются XSL-документы, по умолчанию должен быть указан адрес **http://www.w3c.org/TR/WD-xsl**. Если пространство имен отличается от того, что задано по умолчанию, то процессор, возможно, не выдаст сообщение об ошибке. Однако он будет некорректно осуществлять считывание таблицы стилей и выработает неприемлемые результаты.

- Объявление `default-space` используется в XSL для удаления или сохранения пробелов в XML-документе. Отсутствие этого объявления будет

указывать на то, что пробелы будут сохранены, поскольку это принято по умолчанию для XML.

Элементы, допустимые в элементе *xsl:stylesheet*

В элементе `xsl:stylesheet` разрешается использовать элементы, перечисленные в табл. 7.2.

Таблица 7.2. *Перечень допустимых элементов, входящих в элемент* `xsl:stylesheet`

Элемент	Синтаксис
`xsl:import`	`<xsl:import href="..." />`
`xsl:include`	`<xsl:include href="..." />`
`xsl:strip-space`	`<xsl:strip-space elements="..." />`
`xsl:preserve-space`	`<xsl:preserve-space elements="..." />`
`xsl:key`	`<xsl:key name="..." match="..." use="..." />`
`xsl:functions`	`<xsl:functions ns="..." > ...` `</xsl:functions>`
`xsl:locale`	`<xsl:locale name="...">... </xsl:locale>`
`xsl:attribute-set`	`<xsl:attribute-set name="...">...` `</xsl:attribute-set>`
`xsl:variable`	`<xsl:variable name="...">... </xsl:variable>`
`xsl:param-variable`	`<xsl:param-variable name="...">...` `</xsl:param-variable>`

`xsl:template`	`<xsl:template match="...">...`
	`</xsl:template>`
	`<xsl:template name="...">... </xsl:template>`

Каждый из элементов `xsl:functions`, `xsl:locale`, `xsl:attribute-set`, `xsl:variable`, `xsl:param-variable` и `xsl:template`, представленных в табл. 7.2, может иметь в своем составе другие элементы. Эти элементы должны быть допустимыми узлами исходного дерева в соответствии с табл. 7.1.

Элемент *xsl:template*

Элемент `xsl:template` является основой таблицы стилей XSL. Этот элемент создает шаблон, который позволяет агенту пользователя (user-agent) создавать стилевой итоговый узел из исходного узла.

Этот шаблон в действительности состоит из двух частей: части соответствия (matching) и части обработки (processing).

- Часть соответствия указывает исходный узел, к которому будет применяться часть обработки. Информация этой части содержится в атрибуте. Например, в предыдущей таблице стилей обратите внимание на следующий тег:

```
<xsl:apply-templates select="book_description/edition" />
```

- Часть обработки определяет, как будут обрабатываться дочерние узлы и какое стилевое оформление будет к ним применено. Эта информация содержится в дочерних элементах. Например, в предыдущей таблице стилей обратите внимание на следующий элемент:

```
<xsl:template match="edition">
...
</xsl:template>
```

Соответствия шаблону XSL

Шаблон XSL (XSL template) использует различные маски (patterns) для установления соответствия узлов исходного дерева, как показано в следующих описаниях.

- Простейшим соответствием является соответствие по имени, где исходный элемент легко идентифицируется по его имени, с помощью атрибута `match`. Например, следующему шаблону будут соответствовать все элементы, названные `edition`:

```
<xsl:template match="edition">
```

- В соответствии с "родословной" (ancestry) элемент идентифицируется по имени своего предка, а также по своему собственному имени. Например, указанному ниже шаблону будет соответствовать элемент с именем `Book_Title`, родителем которого является элемент `Book_Description`:

```
<xsl:template match="Book_Description/Book_Title">
```

- Для указания соответствия нескольких имен эти имена должны быть разделены в XSL с помощью символа вертикальной черты `|`, как показано в шаблоне:

```
<xsl:template match="Author | Publisher | Distributor">
```

- Для указания соответствия корню исходного дерева объявление должно иметь вид:

```
<xsl:template match="/">
```

- XSL позволяет устанавливать соответствия, используя идентификатор (ID). Следующий шаблон разрешает найти и установить соответствие для всех узлов исходного дерева с идентификатором, имеющим значение `Important`:

```
<xsl:template match= "id(Important)>
```

- XSL также позволяет устанавливать соответствия, используя атрибуты узлов. Например, если узел `Name` исходного дерева обладает атрибутом `Given`, соответствие шаблону будет иметь следующий вид:

```
<xsl:template match="Name[Given]">
```

Лабораторная работа 7. Создание таблицы стилей XSL

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать таблицу стилей XSL;
- связывать таблицу стилей XSL с XML-документом.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь практические навыки работы с Microsoft Internet Explorer;
- хорошо знать XML.

Сценарий

После создания набора XML-документов вы должны сформировать таблицу стилей для того, чтобы просматривать эти документы в браузере. В данной лабораторной работе вы завершите создание таблицы стилей для отображения страницы, содержащей описание книги, затем вы свяжете XML-документ с таблицей стилей.

Примечание

В этой лабораторной работе в качестве символа продолжения строки используется символ ↳.

Примечание

Время, необходимое для выполнения данной работы, — 15 минут.

Упражнение 1. Создание таблицы стилей XSL

В этом упражнении вы завершите частично созданную таблицу стилей XSL для документа, содержащего описание книги.

Для того чтобы открыть таблицу стилей XSL, необходимо в программе Notepad открыть файл book.xsl из папки <папка_установки>Labs\Lab07\StartCode.

Для объявления таблицы стилей для элемента `price`:

1. В файле book.xsl найдите элемент:

```
<!- Price Template ->
```

2. К разметке, представленной в шаге 1, добавьте следующие строки:

```
<xsl:template match="price">
   <TD BGCOLOR="#FFFFD0" STYLE="width:391;_
   text-align:left;font-family:Arial;font-size:10pt;" >
        <BLOCKQUOTE>
        $<xsl:value-of />
        </BLOCKQUOTE>
   </TD>
</xsl:template>
```

Для того чтобы связать объявленный шаблон с существующим шаблоном:

1. В файле book.xsl найдите разметку:

```
<xsl:apply-templates select="book_description/edition" />
```

2. В строку, следующую за разметкой, представленной в шаге 1, добавьте:

```
<xsl:apply-templates select="book_description/price" />
```

3. Сохраните и закройте файл.

Для того чтобы связать таблицу стилей с данным XML-документом:

1. В программе Notepad откройте файл book.xml из папки <папка_установки>Labs\Lab07\StartCode.
2. Найдите следующее объявление в этом файле:

```
<!DOCTYPE book_description SYSTEM "book.dtd">
```

3. В строку, следующую за объявлением, представленным в шаге 2, добавьте инструкцию для процессора:

```
<?xml-stylesheet type="text/xsl" href="book.xsl" ?>
```

4. Сохраните и закройте файл.

Для просмотра результата в Microsoft Windows NT Explorer перейдите в папку <папка_установки>\Labs\Lab07\StartCode, а затем дважды щелкните мышью на файле book.xml. В браузере будет отображаться итоговая страница, содержащая описание книги.

Вопросы для повторения

1. Какие преимущества по сравнению с другими языками таблиц стилей предоставляются в XSL?
2. Какие есть действительные узлы на исходном дереве?
3. К какому пространству имен, заданному по умолчанию, должна принадлежать таблица стилей XSL?
4. Какое действие по умолчанию выполняется в XSL над пробелами?

ГЛАВА 8

COM-объекты в XML

Краткий обзор

В *главе 5* вы познакомились с созданием COM-компонентов (COM, Component Object Model, модель компонентных объектов) при помощи Microsoft Visual Basic. Традиционно, компоненты COM создавались с помощью таких языков программирования, как Microsoft Visual C++ или Visual Basic. XML позволяет пользователям создавать объекты, именуемые *скриптлетами* (scriptlets), используя для этого скрипт-языки, например, Visual Basic Scripting Edition или Microsoft JScript. Эти скриптлеты являются COM-объектами. Так как они создаются при помощи скрипт-языков, работа по их созданию упрощается.

В этой главе вы познакомитесь со скриптлетами и тем, какое различие существует между DHTML- и XML-скриптлетами. Затем вы узнаете, как создавать и использовать COM-компоненты с помощью XML и Visual Basic Scripting Edition.

Цели и задачи

После прочтения данной главы вы научитесь:

- создавать COM-объект с помощью XML;
- вызывать COM-объект с помощью ASP.

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod07\Mod07Code.txt.

Создание COM-объектов в XML

Далее вы изучите архитектуру XML-скриптлетов и узнаете о различиях, существующих между XML- и DHTML-скриптлетами.

DHTML- и XML-скриптлеты

Скриптлеты (scriptlets) являются HTML-страницами, действующими в качестве динамических компонентов, воздействующих на свойства и методы. Они могут вызывать системные события и выдавать собственные уведомляющие сообщения. Скриптлеты упрощают деятельность Web-разработчиков, предоставляя возможность работать с многократно используемыми компонентами в Web.

Сравнение XML- и DHTML-скриптлетов

И DHTML- и XML-скриптлеты являются компонентами программного обеспечения, самостоятельными, программируемыми, многократно используемыми, независимыми от языка фрагментами кода, которые можно легко встраивать в приложение. Они воздействуют на свойства и методы, инициируют события. Их можно однозначно выявить по имени и/или идентификатору (ID). Отличие между XML- и DHTML-скриптлетами заключается в том, что XML-скриптлеты полностью основываются на моделях компонентных объектов (COM), в то время как в DHTML-скриптлетах используется абсолютно другой подход, который по своим функциональным возможностям близок к возможностям COM. Различия между этими двумя типами скриптлетов представлены в табл. 8.1.

Таблица 8.1. *Сравнение XML- и DHTML-скриптлетов*

Функциональная характеристика	DHTML-скриптлет	XML-скриптлет
Интерфейс программирования (Programming interface)	Посредством функций Microsoft JScript	Библиотека типов
Механизм создания	Функция Microsoft JScript — `public_description`	Функция `DllGetClassObject()`
Методы и свойства	Методы функции `public_description`	Интерфейс автоматизации COM (COM Automation interface)
События	Окно DHTML, внешний объект	Интерфейс указателя соединения COM (COM connection pointer interface)

Таблица 8.1 (окончание)

Функциональная характеристика	DHTML-скриптлет	XML-скриптлет
Язык разработки	HTML и скрипт	VBScript или JScript в XML-документе
Идентификация объекта	Имя файла и MIME-тип	ProgID и CLSID
Регистрация	Не применяется. Имя файла уникальным образом определяет скриптлет	Системный реестр (System Registry)
Размещение в качестве Web-страниц	Тег <OBJECT> и MIME-тип	Тег <OBJECT> и атрибут CLASSID
Размещение в качестве приложения	Посредством управляющего элемента Microsoft ActiveX Control	С помощью непосредственной инсталляции

Архитектура XML-скриптлета

Вы можете написать XML-скриптлет с помощью программного обеспечения разработки на JScript, Microsoft Visual Basic Scripting Edition, Perl, или же на любом другом языке, для которого предусмотрен синтаксический анализатор скриптов с ActiveX. Элементы компонента COM, такие как фабрики

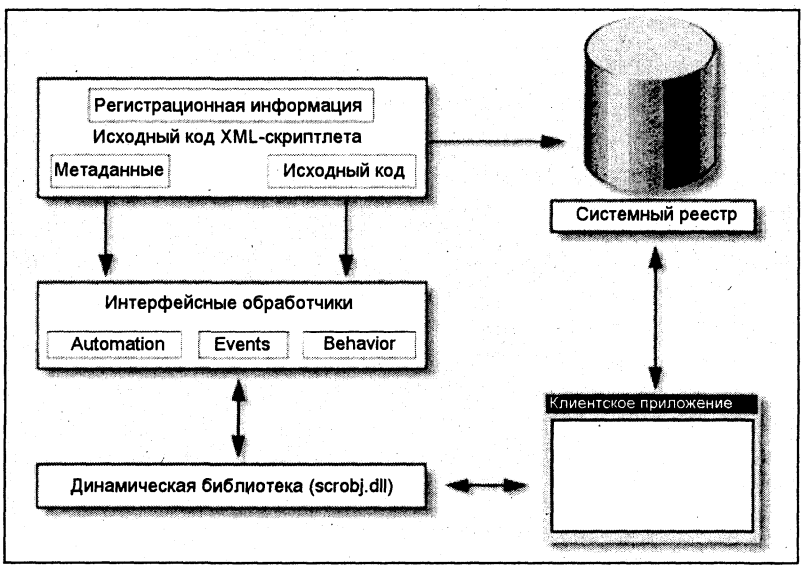


Рис. 8.1. Архитектура XML-скриптлета

классов (class factories), хранятся в системной динамической библиотеке под именем scrobj.dll. Скриптлет использует метаданные для описания объекта во многом таким же способом, как COM использует метаданные для описания объектов. Вам лишь необходимо обеспечить метаданные, которые описывают реализуемые вами интерфейсы, а также код для реализации различных методов.

На рис. 8.1 представлена архитектура XML-скриптлета.

Компоненты XML-скриптлета

XML-скриптлет представляет собой ASCII-файл с расширением sct. В нем имеется три главных тега: `<registration>`, `<implements>` и `<script>`. Весь исходный код XML-скриптлета размещается внутри тега `<scriptlet>`.

- Тег `<registration>` содержит всю информацию, относящуюся к идентификации компонента, такую как `ProgID` и `CLSID`. При разработке XML-скриптлета, описание и номер версии обычно включается в тег `<registration>`, как в следующем примере:

```
<scriptlet>
  <registration
    Description="This is a bare-bones Scriptlet."
    ProgID="BareBones.Scriptlet"
    Version="1.00"
    ClassID="{00000000-1010-1010-83d1-f49604c10010}"
    >
  </registration>
</scriptlet>
```

Примечание

Идентификатор `ClassID` должен быть уникальным, и может быть создан с помощью приложения Guidgen.exe.

- В теге `<implements>` указываются интерфейсы, которые будут реализованы в компоненте. Каждый введенный тег `<implements>` соответствует какому-либо интерфейсу COM, выборку и вызов которого клиенты могут осуществлять посредством `QueryInterface`. Интерфейсы не идентифицируются по их `CLSID` или имени. Идентификация производится с помощью атрибута `type`, содержимое которого играет роль ключевого слова, сообщающего динамическому модулю о вызове COM-сервера для обработки клиентских запросов. У интерфейса может также быть атрибут `id`, который можно использовать в коде как объект. Ниже представлен типичный тег `<implements>`:

```
<implements type="Automation" id="dispatcher">
  <method name="Hello">
```

```
      <parameter name="numHello" />
    </method>
  </implements>
```

- Между тегами <script></script> содержится весь исходный код функций, используемых всеми реализуемыми интерфейсами. Например, в скриптлете, представленном в предыдущем примере, может использоваться тег <script>, в котором реализована выдача строки, состоящей из указанного числа слов "Hello":

```
<script language="JavaScript">
  function Hello (numHello) {
    var strText = "";
    for( i=0; i<numHello; i++)
      strText += strText + "Hello";
    return strText;
  }
</script>
```

Обработчики интерфейсов

Обработчик интерфейсов является исполняемым COM-сервером, который обеспечивает стандартную реализацию заданного интерфейса. Это наиболее важная функциональная возможность XML-скриптлета. В коллекции обработчиков интерфейсов перечислены все интерфейсы COM, которые будут реализованы в скриптлете.

- Обработчики событий позволяют скриптлетам инициировать события. Эти обработчики обеспечивают необходимую инфраструктуру в том, что касается стандартной реализации IConnectionPoint и IConnectionPointContainer.
- Обработчики для автоматизации (Automation handlers) обеспечивают интерфейсы Automation. Они реализуют IDispatchEx. Обработчики событий обеспечивают состыковку интерфейсов, необходимую для запуска событий. Между обработчиками и интерфейсами имеются отношения типа "один-ко-многим".

Ниже приводится пример программного кода генерации события:

```
<script language="JScript">
  function DoWork() {
  MyScriptlet.fireEvent("BeginWork")
    // place code here
  MyScriptlet.fireEvent("WorkCompleted")
  }
</script>
```

Анализ кода при создании COM-объектов в XML

Данные транзакции электронной коммерции введены в интерактивные формы, которые выступают временным хранилищем этих данных. Для постоянного хранения их необходимо записать в базу данных или же в текстовый файл. Далее вы узнаете, как создавать и регистрировать XML-скриптлет, осуществляющий запись данных из интерактивной формы в текстовый файл. Вы также узнаете, как использовать скриптлет на ASP-странице.

XML-скриптлет

В транзакции электронной коммерции должна быть предусмотрена возможность идентификации пользователя, с которым осуществляется взаимодействие. Вам также может потребоваться просмотреть историю последней транзакции пользователя. Для выполнения этих задач прежде всего необходимо записать данные о пользователе (user's credentials).

Вы узнаете, как создать скриптлет, который считывает из формы имя, адрес, номер кредитной карточки пользователя, а затем сохраняет эту информацию в текстовом файле, созданном скриптлетом, на диске C:.

```
<scriptlet>
  <registration
    Description="This scriptlet is used to write data from _
    a form to a text file"
    <!- Description="Этот скриптлет используется для записи ↳
    данных из формы в текстовый файл" ->
    ProgID="Storage.Scriptlet"
    Version="1.0"
    ClassId="{8D9CC880-D79F-11d2-B7C8-00C0DFE39737}"
  >
  </registration>
  <implements type="ASP" />
  <public>
    <method name="whoAreYou" />
  </public>
  <script language="VBSCript">
    public function whoAreYou(who, where, number)
      dim usr_name, usr_address, usr_credit
      usr_name = who
      usr_address = where
      usr_credit = number
      Set fso = CreateObject("Scripting.FileSystemObject")
      Set MyFile = fso.CreateTextFile("c:\credibility.txt", True)
```

```
      MyFile.WriteLine(usr_name)
      MyFile.WriteLine(usr_address)
      MyFile.WriteLine(usr_credit)
      MyFile.Close
    end function
  </script>
</scriptlet>
```

Тег <registration>

Тег `<registration>`, показанный в предыдущем примере, имеет следующие атрибуты.

- `Description` (описание). Используется для описания скриптлета, чтобы тот, кто просматривает sct-файл мог определить предоставляемые скриптлетом функциональные возможности.
- `ProgID`. Служит для вызова скриптлета.
- `Version`. Применяется для идентификации версии скриптлета.
- `ClassID`. Предназначен для регистрации скриптлета на том компьютере, на котором он будет выполняться.

Тег <implements>

Тег `<implements>` содержит следующие объявления:

- скриптлет, который будет вызываться в ASP-файле;
- указание, что данный скриптлет поддерживает метод `whoAreYou`.

Тег <script>

Тег `<script>` указывает, что:

- метод `whoAreYou` принимает три параметра: `who`, `where` и `number`.
- данный скриптлет создает файл credibility.txt в корневой папке диска C: компьютера и записывает данные, переданные ему, в этот файл.

После создания скриптлета его необходимо зарегистрировать на том компьютере, на котором он будет выполняться. Для этого необходимо в Windows NT Explorer щелкнуть правой кнопкой мыши на файле скриптлета, а затем выбрать команду **Register**. Это приведет к регистрации скриптлета на компьютере в качестве COM-объекта.

Примечание

Подробная информация о регистрации компонентов COM содержится в MSDN Library.

После регистрации скриптлета его можно вызвать как объект, используя ASP-файл. Образец ASP-файла, предназначенный для вызова скриптлета, выглядит следующим образом:

```
<%
  Set oData = CreateObject("Storage.Scriptlet")
  rs=oData.tell(request.form("name"),request.form("address"), _
  request.form("phone"))
%>
```

Примечание

Для более детального ознакомления с XML-скриптлетами *см. приложение С.*

Лабораторная работа 8. Создание и работа с COM-объектами в XML

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать и регистрировать COM-объект в XML;
- вызывать COM-объект с помощью ASP-страницы.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь знания HTML и технологии активных серверных страниц (ASP, Microsoft Active Server Pages);
- уметь работать с Microsoft Visual Basic Scripting Edition;
- иметь базовые знания о модели компонентных объектов (COM, Component Object Model);
- иметь практические навыки работы с Microsoft Internet Explorer.

Подготовка к работе

Для выполнения этой работы необходимо скопировать файлы из папки Lab\Lab08\StartCode в папку C:\InetPub\wwwroot.

Сценарий

После того как пользователь сделает заказ на закупку, данный заказ должен быть выполнен. Это подразумевает необходимость считывания введенных

пользователем данных и отправку содержания необходимых полей в файл. В этой лабораторной работе в качестве полей, содержание которых должно быть передано в заказ на закупку, вы будете использовать `item code` (код номенклатуры), `item name` (наименование номенклатуры заказа) и `quantity ordered` (заказанное количество).

Примечание

Время, необходимое для выполнения данной работы, — 15 минут.

Упражнение 1. Создание и регистрация COM-компонента в XML

В этом упражнении вы создадите компонент COM в XML и зарегистрируете его.

Для включения регистрационной информации в скриптлет:

1. В программе Notepad откройте файл purchase_order.sct, расположенный в папке C:\InetPub\wwwroot.
2. Найдите тег `<registration>`, в котором после слова `registration` добавьте следующее:

```
Description="This scriptlet is used to write a purchase order from
  form input"
ProgID="Purchase_order.Scriptlet"
Version="1.0"
```

3. Сохраните данный файл.

Для включения `ClassID` в скриптлет:

1. В меню кнопки **Start** выберите команду **Run**.
2. Перейдите в папку C:\InetPub\wwwroot, а затем дважды щелкните по файлу Guidgen.exe.
3. Для выполнения программы нажмите кнопку **OK**.
4. В окне **Create GUID** (Создать GUID) щелкните **Registry Format** (Формат реестра).
5. Выберите **New GUID** (Новый GUID), и затем щелкните по кнопке **Copy** для генерации нового идентификатора `ClassID` и копирования его в буфер обмена.
6. Нажмите кнопку **Exit** для того, чтобы закрыть окно **Create GUID**.
7. В файле purchase_order.sct найдите параметр `classid=""`, а затем установите курсор между кавычками.

8. В меню **Edit** выберите команду **Paste**, для того чтобы вставить `ClassID` в скриптлет.

Для того чтобы написать скрипт, который создает объект:

1. Найдите теги `<script language = "VBScript">` и `</script>`. Между ними вставьте следующий код:

```
public function hello(code, name, howmany)
       dim pur_code, pur_nam, pur_quant
       pur_code = code
       pur_nam = name
       pur_quant = howmany
       Set fso = CreateObject("Scripting.FileSystemObject")
       Set MyFile = fso.CreateTextFile("c:\purchase_order.txt", True)
       MyFile.WriteLine(pur_code)
       MyFile.WriteLine(pur_nam)
       MyFile.WriteLine(pur_quant)
       MyFile.Close
end function
```

2. Сохраните и закройте данный файл.

Для того чтобы зарегистрировать COM-объект:

1. В меню кнопки **Start** укажите пункт **Programs**, а затем выберите команду **Windows NT Explorer**.
2. Просмотрите папку C:\InetPub\wwwroot, а затем щелкните правой кнопкой мыши на файле purchase_order.sct.
3. В контекстном меню выберите команду **Register**.
4. Отобразится окно сообщения об успешной регистрации компонента.

Упражнение 2. Вызов COM-объекта с помощью ASP-страницы

В этом упражнении вы создадите ASP-страницу, предназначенную для считывания данных из формы ввода, и примените COM-объект для генерации заказа на закупку.

Для того чтобы открыть форму, предназначенную для ввода значений переменных:

1. Запустите Microsoft Internet Explorer, и затем в поле **Address** введите следующий URL: **http://localhost/Purchase_order.htm**.
2. В меню **View** выберите команду **Source**. Обратите внимание на переменные, используемые для ввода на форме: **Item Code**, **Item Name** и **Quantity**.
3. Закройте данный файл.

Для того чтобы сформировать ASP-страницу:

1. В программе Notepad создайте новый файл.
2. В этом файле введите следующий код:

```
<%
Set oData = CreateObject("Purchase_order.Scriptlet")
            rs=oData.hello(request.form("item_code"),request.form _
           ("item_name"),request.form("quantity"))
            response.write("Ваш заказ принят. " & _
           "Мы скоро его вам доставим")
%>
```

3. Сохраните данный файл как purchase_order.asp в папке C:\InetPub \wwwroot, а затем закройте его.

Для просмотра выводимых данных:

1. В Microsoft Internet Explorer в поле **Address** введите следующий URL: **http://localhost/purchase_order.htm**.
2. На странице purchase_order.htm введите данные:
   - в поле **Item Code** (Код номенклатуры товара) введите `ST01`;
   - в поле **Item Name** (Наименование товара) введите `Pencils`;
   - в поле **Quantity** введите `15`.
3. Нажмите кнопку **Submit Form** (Отправить форму).
4. Будет отображаться страница с сообщением "Ваш заказ принят. Мы скоро его вам доставим".
5. В Windows NT Explorer перейдите на C:\purchase_order.txt, а затем дважды щелкните по файлу для просмотра его содержимого.
6. Появятся значения, указанные на странице purchase_order.htm.

Вопросы для повторения

1. Какие три тега должны быть в XML-скриптлетах?
2. В чем преимущества написания COM-объектов в XML?
3. Какой динамический файл используют по умолчанию XML-скриптлеты?
4. В чем заключается разница интерфейсов программирования DHTML- и XML-скриптлетов?

ГЛАВА 9

Электронный обмен данными EDI

Краткий обзор

Электронный обмен данными (EDI, Electronic Data Interchange) — это электронная передача или обмен данными между организациями. Хотя детальное описание EDI выходит за рамки настоящей главы, вы познакомитесь с основами EDI и его преимуществами.

Даже если EDI предлагает очевидные преимущества, не все компании реализуют эту форму обмена данными. В результате, они часто испытывают необходимость в преобразовании данных, представленных в различных форматах — в EDI-формат. Один из методов такого преобразования заключается в применении ASP-страниц, точно таким же образом, как использовался компонент `MakePO` CIP-конвейера. Однако самый простой способ преобразования данных в наборы транзакций EDI заключается в использовании XML. В заключительном разделе этой главы вы узнаете о том, как конвертировать данные из XML в EDI.

Цели и задачи

После прочтения данной главы вы научитесь:

- формулировать определение EDI и перечислять его преимущества;
- конвертировать XML-документ в набор транзакций EDI (EDI transaction set).

Примечание

Код, представленный в этой главе, размещен на CD-ROM в файле DemoCode\Mod09\Mod09Code.txt.

Обзор EDI

EDI представляет собой межкомпьютерный обмен обычной бизнес-информацией в стандартном формате. Так как EDI позволяет компаниям осуществлять практически мгновенный обмен информацией, он сводит к минимуму бумажную работу и делает выполнение внутренних операций более эффективным, отвечающим нуждам клиентов.

Что такое EDI?

EDI или электронный обмен данными является общепризнанным стандартом обмена данными между торговыми партнерами (рис. 9.1).

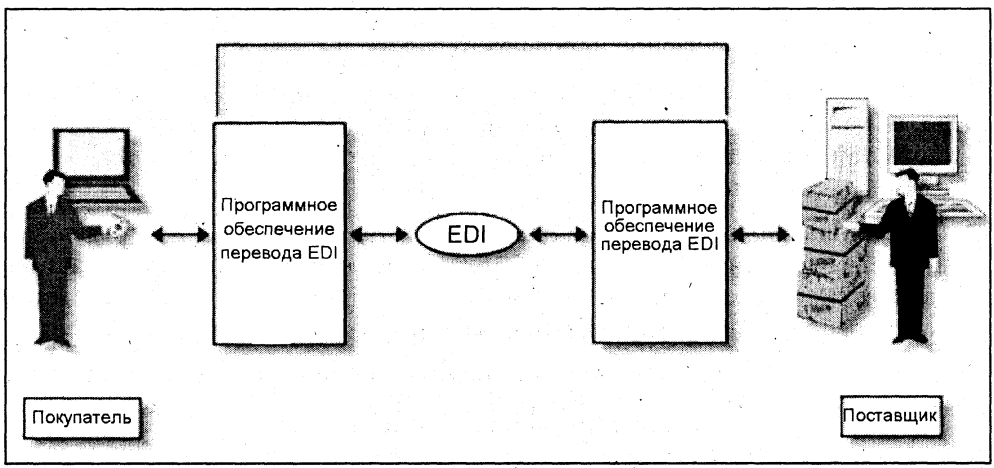


Рис. 9.1. EDI

EDI определяет форматы, типы данных, а также инструкции по маршрутизации для электронного обмена бизнес-документами между компьютерными системами различных компаний. Эти транзакции обычно осуществляются в частных сетях с дополнительными услугами (VAN, Value Added Network). EDI сводит к минимуму массу бумажных заказов на закупку, счетов, транспортных и других документов, и заменяет их электронными аналогами. Это позволяет сократить издержки обработки документации в полтора раза.

Термины

Среди терминов, используемых в EDI, есть такие, как *торговые партнеры* (trading partners), *электронные конверты* (electronic envelopes), а также *программное обеспечение перевода EDI* (EDI translation software). Каждый из этих терминов описан в следующих разделах.

Торговые партнеры

В терминологии EDI торговый партнер представляет собой некоторую единицу (или сущность), осуществляющую бизнес-операции по сети. В качестве такой сущности может выступать компания, агентство или заказчик. Все торговые партнеры подписывают торговое партнерское соглашение, что делает их EDI-транзакции законными.

Электронные конверты

Информация распространяется в Web в так называемых электронных конвертах. При получении электронного конверта компьютер ищет специальный ключ, предназначенный для идентификации присланной информации. Ключ состоит из следующих элементов.

- ISA-объявление (Instrument Society of America) указывает начало данного конверта. IEA-объявление определяет его конец.
- Конверт может состоять из одной и более функциональных групп. Начало и конец каждой функциональной группы задается, соответственно, с помощью признака начала группы — GS (group start) и признака конца группы — GE (group end).
- Функциональные группы хранят наборы транзакций EDI. Объявление заголовка набора транзакции, ST, указывает начало набора транзакции, являющейся эквивалентом бумажного документа. SE, или объявление конца набора транзакции, задает конец набора транзакции.

Ниже представлен пример электронного конверта. Обратите внимание на объявления ISA, IEA, GS, GE, ST и SE в этом электронном конверте.

```
ISA*00*00*01*0860603544*08*6113240000*940608*0902*U*00303*00K
0000451*0*P>~
  GS*IN*086063544*6113240000*940608*0902*1156*X*003030~
    ST*850*6541~
      BEG*00*SA*09935600*990707~
      NTE**GOODS SHIPPED PER EDI INSTRUCTIONS~
      REF*1A*053445~
      CSH*P2~
      DTM*010*990707~
      DTM*001*990707~
      TD5*****1-200 COPIES~
      TD5*****201+ MICROSOFT~
      N1*BY*92*08995~
      PID*F*99***2T - 4T~
    SE*15*6541~
  GE*1*1156~
IEA*1*000451~
```

Примечание

Символы * в предыдущем коде являются символами-заполнителями. Код представлен с отступами, для удобства зрительного восприятия. В реальном наборе EDI-транзакции отступы отсутствуют.

Программное обеспечение перевода EDI

Программное обеспечение перевода EDI (EDI translation software) осуществляет конвертирование данных из исходного формата в набор транзакции EDI перед отправкой в сеть. При получении набора транзакции EDI программное обеспечение перевода EDI производит обратное конвертирование данных в исходный формат.

Преимущества EDI

EDI позволяет упростить основной поток транзакций между торговыми партнерами, путем замены всех бумажных документов их электронными эквивалентами. Так как EDI-документы все передаются электронным способом, то снижаются почтовые издержки и время заказа. EDI гарантирует более высокую надежность данных, поскольку данные вводятся лишь один раз, а потом многократно используются по необходимости.

На рис. 9.2 показан поток информации в бизнес-транзакции EDI.

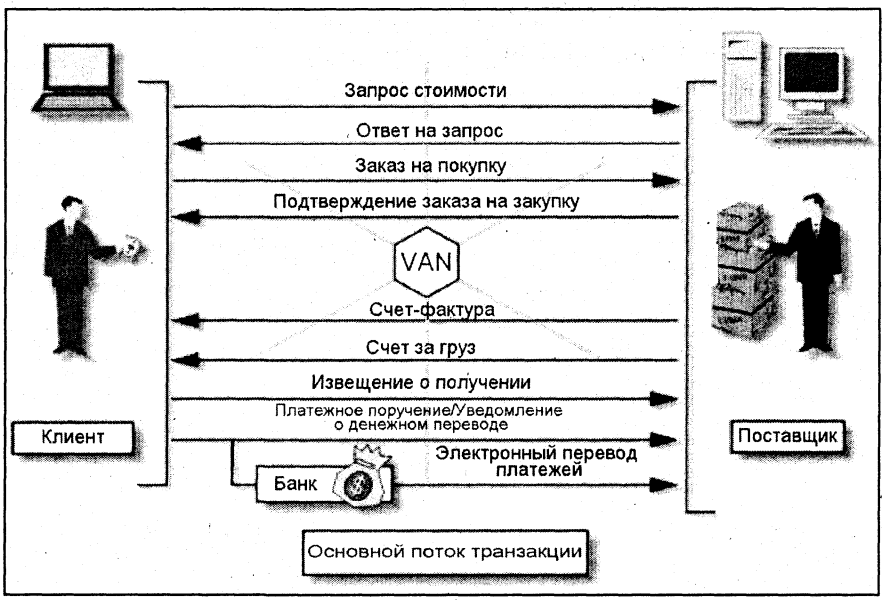


Рис. 9.2. Поток информации в бизнес-транзакции EDI

Основной поток транзакции

Ниже представлена последовательность для типичного примера потока бизнес-транзакции EDI:

1. Заказчик отправляет поставщику запрос о стоимости и техническую спецификацию на продукцию.
2. Поставщик отвечает на этот запрос.
3. Заказчик производит оценку стоимости и затем отправляет заказ на закупку поставщику.
4. Когда продукция будет готова к отгрузке, поставщик посылает заказчику счет-фактуру или транспортную накладную (advance ship notice).
5. Продукция отгружена. Поставщик посылает счет за груз (freight bill) заказчику.
6. Когда заказчик получает продукцию, поставщику отправляется уведомление о получении.
7. Заказчик отправляет платеж.

На каждом шаге получатель посылает отправителю квитанцию о подтверждении получения соответствующей информации, что приводит к удорожанию процесса, времени двустороннего обмена, а также к росту риска ошибки. EDI значительно упрощает этот процесс.

Результаты использования EDI в основном потоке транзакции

Упрощая основной поток транзакции, EDI предоставляет следующие преимущества.

- В сценарии электронной коммерции EDI ускоряет выполнение цикла "заказ на закупку — платеж по счету", обеспечивая тем самым более быстрое продвижение товаров на рынок сбыта.
- EDI предлагает обмен электронными формами документов, тем самым снижая издержки на создание бумажных документов, их почтовую рассылку и регистрацию переписки.
- В EDI данные необходимо вводить только один раз. Затем эти данные могут использоваться многократно. Это исключает необходимость повторного ввода одних и тех же данных, снижает вероятность возникновения ошибок.
- В EDI информация обновляется быстрее, что в свою очередь способствует более быстрому принятию решений.
- EDI позволяет поставщикам отправлять лишь то, что необходимо, сокращая запасы у дистрибьюторов (distributors' inventory), отправляя по-

ставщикам информацию о торговых точках. К тому же, благодаря сокращению времени, требуемого на выполнение заказа, EDI способствует тому, что дистрибьюторы работают с меньшими запасами.

Примечание

Для получения более подробной информации об EDI обратитесь к Web-сайту DISA по адресу **http://www.disa.org/**.

XML и EDI

Хотя EDI предлагает значительную выгоду, стоимость реализации очень высока. Поэтому EDI используют только большие компании. Если компания, в которой EDI не используется, захочет взаимодействовать с такой компанией, в которой эта технология реализована, ей потребуется конвертировать свои данные в наборы транзакций EDI. И наоборот, данные, которые эта компания получает, должны быть преобразованы из формата EDI в тот формат, который используется в компании, принимающей данные. Самый простой способ для этого заключается в хранении данных в XML-документах. XML позволяет структурировать данные, придавая им необходимую гибкость и наиболее удобный вид для преобразования в структуры, согласующиеся с EDI.

Конвертирование данных из XML в EDI

Данные, которые традиционно используются в виде бумажных документов, например заказы по счетам, могут быть преобразованы в XML с помощью ASP-страниц. Данные, которые затем будут представлены в этом формате, будут удобны как для зрительного восприятия, так и для машинной обработки.

Преобразование XML-документа в EDI

Типичный счет содержит подробную информацию о покупке, покупателе, и организации-предъявителе данного счета. В XML это может быть представлено следующим образом:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="801.xsl" ?>
<invoice number="1045">
   <organization status="send to">
      <name>RAMONA PUBLISHING</name>
      <street>P.O. BOX 679342</street>
      <city>HOUSTON</city>
      <state>TX</state>
```

```
   <zip>77234</zip>
</organization>
<organization status="charge">
   <name>HANSEL PUBLISHING</name>
   <street>101 APPLE PIE ST.</street>
   <city>NEW YORK</city>
   <state>NY</state>
   <zip>00103</zip>
</organization>
<organization status="sent from">
   <name>FIVELAKES PUBLISHING</name>
   <street>79 RIVER DRIVE</street>
   <city>DALLAS</city>
   <state>TX</state>
   <zip>74564</zip>
</organization>
<shipper>
   <name>CONSOLIDATED TRUCK</name>
   <street>534 FLAT MEADOWS</street>
   <city>HOUSTON</city>
   <state>TX</state>
   <zip>77544</zip>
</shipper>
<contact>
   <name>J. DOE</name>
   <phone>2104355445</phone>
</contact>
<order>
   <item>
      <code>6900</code>
      <quantity>3</quantity>
      <unit>CA</unit>
      <price_per_unit>12.75</price_per_unit>
   </item>
   <item>
      <code>P450</code>
      <quantity>12</quantity>
      <unit>EA</unit>
      <price_per_unit>2.99</price_per_unit>
   </item>
   <item>
      <code>1640</code>
      <quantity>4</quantity>
      <unit>EA</unit>
      <price_per_unit>5.99</price_per_unit>
```

```
      </item>
      <item>
         <code>1507</code>
         <quantity>1</quantity>
         <unit>DZ</unit>
         <price_per_unit>2.45</price_per_unit>
      </item>
      <total_cost>100.54</total_cost>
      <date>
         <day>20</day>
         <month>4</month>
      </date>
   </order>
</invoice
```

С помощью XSL предыдущий XML-код может быть легко преобразован в транзакцию EDI. Ниже представлен образец таблицы стилей:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
   <xsl:template match="/">
      ISA**00*0000000000*01*01*PASSWORDME*01*123456789 ↳
      987654321 890714*2210*U*000000008*O*P*~
      GS*IN*012345678*087654321*900509*2210*000001*X*002040~
      ST*801*0001~74832
      BEG*940606*1045*940606*~
      <xsl:apply-templates select="invoice/organization" />
      <xsl:apply-templates select="invoice/contact" />
      ITD*01*3*2**10~
      <xsl:apply-templates select="invoice/order/item" />TDS*
      <xsl:value-of select="invoice/order/total_cost" />~
      <xsl:apply-templates select="invoice/shipper" />
      <xsl:apply-templates select="invoice/order/date" />
      SE*21*000001~
      GE*1*000001~
      IEA*1*000000008~
   </xsl:template>
   <xsl:template name="org" match="organization">
      N1*BT*
      <xsl:value-of select="name" />~ N3*
      <xsl:value-of select="street" />~ N4*
      <xsl:value-of select="state" />*
      <xsl:value-of select="zip" />~
   </xsl:template>
   <xsl:template match="contact">
      PER*AD*
      <xsl:value-of select="name" />*TE*
```

```
      <xsl:value-of select="phone" />~
   </xsl:template>
   <xsl:template match="item">
      IT1**
      <xsl:value-of select="quantity" />**VC**
      <xsl:value-of select="code" />~
   </xsl:template>
   <xsl:template match="shipper">
      CAD*M****
      <xsl:value-of select="name" />~
   </xsl:template>
   <xsl:template match="date">
      CTT*
      <xsl:value-of select="month" />*
      <xsl:value-of select="day" />~
   </xsl:template>
</xsl:stylesheet>
```

Анализ таблицы стилей

Таблица стилей вызывается в XML-документе и служит для конвертирования XML-данных в формат для представления счета. Этот формат определен спецификацией ANSI ASC X12. Субстили определены в корневом разделе данной таблицы стилей. Каждый субстиль (substyle) выбирает требуемые данные и передает необходимые для EDI объявления, делая данные удобными для чтения.

Конвертирование набора транзакций EDI в XML

Хотя с помощью таблицы стилей конвертирование из XML в EDI осуществляется легко, обратный процесс — конвертирование из EDI в XML — может оказаться проблематичным. Наиболее удобный способ для этого заключается в использовании синтаксического анализатора EDI (EDI parser) для конвертирования EDI. Этот анализатор может быть написан на таком языке высокого уровня, как Microsoft Visual C++ или Microsoft Visual Basic, или же с помощью XML в скрипт-языке. Следующий пример демонстрирует XML-скриптлет, который используется для конвертирования счета из набора транзакции EDI 801 в XML-документ:

```
public sub makeXML()

  Const ForReading = 1
  Const TristateUseDefault = -2, TristateTrue = -1, _
  TristateFalse = 0

  Dim fs, f, ts, s
  Set fs = CreateObject("Scripting.FileSystemObject")
```

```
  Set f = fs.GetFile("C:\edi.txt")
  Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
  Storage = ts.ReadAll
  ts.Close

  Dim num, I
  num = InStr(1, Storage, "N1*BT*")
  num = num - 1
  t = Left(Storage, num)
  Storage = Replace(Storage, t, " ")
  leng1 = Len(Storage)
  num = InStr(1, Storage, "SE*")
  num = leng1 - num + 1
  t = Right(Storage, num)
  Storage = Replace(Storage, t, " ")
  new1 = "<organization>" & Chr(13) & "<name>"
  Storage = Replace(Storage, "N1*BT*", new1)
  new1 = "</name>" & Chr(13) & "<street>"
  Storage = Replace(Storage, "~ N3*", new1)
  ...
  Storage = Replace(Storage, "0<item>", new1)
  new1 = "</quantity>" & Chr(13) & "<code>"
  Storage = Replace(Storage, "*VC* ", new1)
  Storage = Replace(Storage, "*", ",")
  Storage = Replace(Storage, "~", "")
  Set fs = CreateObject("Scripting.FileSystemObject")
  Set MyFile = fs.CreateTextFile("c:\edi-new.xml", True)
  MyFile.WriteLine ("<?xml version='1.0' ?>")
  MyFile.WriteLine ("<invoice>")
  MyFile.Write (Storage)
  MyFile.Close
End Sub
```

Анализ скрипта

Набор транзакций EDI хранится в текстовом файле edi.txt на диске C:. Данный скриптлет открывает этот файл и считывает его содержимое в переменную `Storage` с помощью следующего кода:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile("C:\edi.txt")
Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
Storage = ts.ReadAll
```

Вся последующая обработка выполняется над значениями, хранящимися в этой переменной. Соответствующие разделы набора транзакции идентифицируются и замещаются тегами XML с помощью следующего кода:

```
newl = "<organization>" & Chr(13) & "<name>"
Storage = Replace(Storage, "N1*BT*", newl)
```

После того как значения этой переменной будут модифицированы для получения действительных XML-данных, эти данные записываются в файл edi-new.xml на диске C: с помощью следующего кода:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set MyFile = fs.CreateTextFile("c:\edi-new.xml", True)
MyFile.WriteLine ("<?xml version='1.0' ?>")
MyFile.WriteLine ("<invoice>")
MyFile.Write (Storage)
```

Весь код размещается в теге `<script>` XML-скриптлета, который может быть зарегистрирован на сервере и вызван с помощью ASP-страницы.

Лабораторная работа 9. Преобразование XML-документа в EDI

Цели

После выполнения этой лабораторной работы вы должны уметь:

- формировать таблицу стилей XSL для конвертирования XML-документа в набор транзакций EDI;
- создавать XML-скриптлет для конвертирования набора транзакции EDI в XML-документ.

Перед началом работы

Предварительные требования

Перед началом работы необходимо хорошо знать:

- XSL;
- XML;
- Microsoft Visual Basic Scripting Edition;
- Microsoft Internet Explorer.

Подготовка к лабораторной работе

Для выполнения этой лабораторной работы вы должны скопировать файлы, находящиеся в папке Lab\Lab09\StartCode в папку C:\InetPub\wwwroot.

Сценарий

В данной лабораторной работе воспроизведена ситуация, которая возникает, когда не использующее EDI предприятие взаимодействует с компанией, в которой применяется EDI. Сначала вам необходимо сформировать таблицу стилей XSL, которую вы будете использовать для конвертирования XML-документа счета в набор транзакции EDI 801. Затем вы напишете XML-скриптлет, предназначенный для обратного преобразования набора транзакций EDI в XML.

Примечание

Время, необходимое для выполнения данной работы, — 30 минут.

Упражнение 1. Создание таблицы стилей

В этом упражнении вы завершите создание таблицы стилей XSL, которую вы будете использовать для преобразования XML-документа в набор транзакций EDI.

Для модификации таблицы стилей XSL в папке C:\InetPub\wwwroot откройте файл edi.xsl с помощью программы Notepad.

Для определения шаблона, соответствующего `organization`, в таблице стилей XSL:

1. В файле edi.xsl найдите следующую разметку:

```
<xsl:template name="who" match="organization">
```

2. В строку, следующую за кодом, представленным в шаге 1, вставьте код для таблицы стилей:

```
N1*BT*
<xsl:value-of select="name" />~ N3*
<xsl:value-of select="street" />~ N4*
<xsl:value-of select="state" />*
<xsl:value-of select="zip" />~
```

3. Сохраните данный файл.

Для связывания шаблона в таблице стилей:

1. В файле edi.xsl найдите текст:

```
BEG*940606*1045*940606*~
```

2. В строке, следующей за текстом, представленным в шаге 1, добавьте разметку:

```
<xsl:apply-templates select="invoice/organization" />
```

3. Сохраните и закройте данный файл.

Упражнение 2. Связывание таблицы стилей с XML-документом

В этом упражнении вы свяжите таблицу стилей XSL, созданную вами в упражнении 1, с XML-документом.

Для того чтобы открыть XML-документ, в папке C:\InetPub\wwwroot откройте файл edi.xml с помощью программы Notepad.

Для связи таблицы стилей XSL с XML-документом:

1. Найдите в данном документе разметку:

```
<?xml version="1.0" ?>
```

2. В строке, следующей за разметкой, представленной в шаге 1, добавьте разметку:

```
<?xml-stylesheet type="text/xsl" href="edi.xsl" ?>
```

3. Сохраните и закройте данный файл.

Для просмотра результата в Microsoft Windows NT Explorer перейдите в папку C:\InetPub\wwwroot и дважды щелкните мышью на файле edi.xml. В браузере будет отображен набор транзакции EDI 801.

Упражнение 3. Создание XML-скриптлета

В этом упражнении вы завершите XML-скриптлет, который будете использовать для конвертирования набора транзакции EDI XML-документа.

Для открытия XML-скриптлета в папке C:\InetPub\wwwroot откройте файл edi.sct с помощью программы Notepad.

Для включения регистрационной информации в скриптлет:

1. Найдите тег `<registration>` и после слова `registration` добавьте следующее:

```
Description="This scriptlet is used to convert a 801 _
Transaction Set into XML"
ProgID="xmliser.Scriptlet"
Version="1.0"
```

2. В меню кнопки **Start** выберите команду **Run**.

3. Перейдите в папку C:\InetPub\wwwroot, дважды щелкните на файле Guidgen.exe, а затем нажмите кнопку **OK** для выполнения программы.

4. В окне **Create GUID** (Создать GUID) щелкните **Registry Format** (Формат реестра), а затем нажмите **New GuiD** для генерации уникального идентификатора `ClassID`.

5. Щелкните по кнопке **Copy** для того, чтобы поместить `ClassID` в буфер обмена, а затем нажмите кнопку **Exit**.
6. В теге `<registration>` в файле edi.sct найдите вхождение `classid=""`, а затем вставьте между кавычками скопированный вами в буфер обмена идентификатор `ClassID`.
7. Сохраните этот файл.

Для записи скрипта, выполняющего скриптлет:

1. В файле edi.sct найдите:

```
Const ForReading = 1
Const TristateUseDefault = -2, TristateTrue = -1, _
TristateFalse = 0
Dim fs, f, ts, s
```

2. За фрагментом, представленным в шаге 1, вставьте следующий код, предназначенный для чтения набора транзакции EDI с диска C: в переменную:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set f = fs.GetFile("C:\Inetpub\wwwroot\edi.txt")
Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
Storage = ts.ReadAll
ts.Close
```

3. В файле edi.sct найдите:

```
lengl = Len(Storage)
```

4. Следом за кодом, представленным в шаге 3, добавьте следующий код, предназначенный для удаления завершающей информации EDI (trailing EDI information), которая не нужна для счета:

```
num = InStr(1, Storage, "SE*")
num = lengl - num + 1
t = Right(Storage, num)
Storage = Replace(Storage, t, " ")
```

5. В файле edi.sct найдите код:

```
Storage = Replace(Storage, "*VC* ", newl)
Storage = Replace(Storage, "*", ",")
Storage = Replace(Storage, "~", "")
```

6. В строке, следующей за кодом, представленным в шаге 5, введите код, служащий для записи XML-документа:

```
Set fs = CreateObject("Scripting.FileSystemObject")
Set MyFile = fs.CreateTextFile("c:\edi-new.xml", True)
```

```
MyFile.WriteLine ("<?xml version='1.0' ?>")
MyFile.WriteLine ("<invoice>")
MyFile.Write (Storage)
MyFile.Close
```

7. Сохраните и закройте данный файл.

Для регистрации компонента:

1. В меню кнопки **Start** укажите пункт **Programs**, а затем выберите команду **Windows NT Explorer**.
2. Перейдите в папку C:\InetPub\wwwroot, а затем щелкните правой кнопкой мыши на файле edi.sct.
3. В контекстном меню выберите команду **Register**.
4. Появится окно сообщений, указывающее на то, что данный скриптлет был успешно зарегистрирован как COM-объект.

Упражнение 4. Вызов скриптлета с помощью ASP-страницы

В этом упражнении вы создадите ASP-страницу, в которой используется созданный вами в упражнении 3 скриптлет для конвертирования набора транзакции EDI в XML-документ.

Для создания необходимой ASP-страницы:

1. В программе Notepad создайте новый файл.
2. В этот файл добавьте следующий код:

```
<SCRIPT Language ="VBScript" RUNAT="Server">
   set Change = CreateObject("xmliser.Scriptlet")
   Change.makeXML()
   Response.Write("Преобразование в XML завершено")
</SCRIPT>
```

3. Сохраните данный файл как edi.asp в папке C:\InetPub\wwwroot, а затем закройте его.

Для просмотра выводимых данных:

1. В Microsoft Internet Explorer в поле **Address** введите URL: **http://localhost/edi.asp**. Появится страница с сообщением: "Преобразование в XML завершено".
2. В Windows NT Explorer перейдите к файлу C:\edi-new.xml, а затем дважды щелкните по нему для просмотра его содержимого. В Microsoft Internet Explorer появится данный XML-документ.

Вопросы для повторения

1. Какие компоненты имеет электронный конверт?
2. Каковы преимущества использования EDI?
3. Почему компаниям необходимо преобразовывать данные, представленные в различных форматах, в EDI?
4. Почему XSL нельзя использовать для преобразования данных из EDI в другие форматы?

ГЛАВА 10

Сохранение данных о состоянии сеанса

Краткий обзор

В этой главе вы узнаете, как сохраняются *данные о состоянии сеанса* (session state data) на сайте при помощи *объекта активного пользователя* (AUO, Active User Object) и *каталога принадлежности* (Membership Directory).

По окончании работы над этой главой вы сможете:

- настраивать AUO для сайта;
- редактировать ASP-страницы таким образом, чтобы они могли записывать данные состояния сеанса.

Введение в сохранение данных состояния сеанса

В этой главе представлены сведения:

- о необходимости сохранения данных состояния сеанса;
- о методах сохранения данных состояния сеанса.

Необходимость сохранения данных

Данные сеанса (session data) — это информация о пользователе или клиенте сайта, которую вы хотите записать и отследить. Данная информация полезна по ряду причин, поскольку позволяет:

- проверить, посещал ли пользователь рекламную страницу (promotional page) на сайте. Если пользователь еще не посещал эту страницу, то тогда можно отобразить ссылку на нее в текущей странице;

- отобразить учетные ставки, приемлемые для того бизнес-партнера, который только что посетил данный сайт;
- отследить количество заказов, инициированных бизнес-партнером;
- отобразить те заказы, которые бизнес-партнер сделал в течение определенного периода времени;
- отобразить список платежных планов (list of payment), предоставленных пользователем в определенный период времени;
- в зависимости от времени, когда пользователь оплачивает заказанные товары, вычислить учетный процент (учетную ставку). Это можно применить к годовым платежным планам, которые вы, вероятно, будете предлагать вашим бизнес-партнерам;
- отобразить список всех клиентов, посетивших сайт или текущую страницу;
- поместить на странице соответствующий код, который выполнится в том случае, когда пользователь обращается к странице более определенного количества раз. Например, на вашем сайте могут отображаться данные фондовой биржи. Если посетитель сайта перейдет на данную страницу свыше двух раз за один сеанс, может произойти переключение на ту часть кода, которая отобразит страницы, специально относящиеся к фондовой бирже;
- отобразить список всех изделий, которые пользователь отметил в текущей сессии, но не поместил в покупательскую корзину;
- проанализировать использование сайта и шаблонов покупок (buying patterns) заказчиков, которые посетили ваш сайт, с помощью Site Server Analysis Tool;
- отобразить список предыдущих транзакций заказчика.

Вы можете также сохранить данные сеанса для дальнейшего их использования, например, для того, чтобы тем пользователям, которые часто посещают сайт, предоставлять персонифицированные страницы. Это помогает создать набор параметров (профиль) для пользователя и записать пользовательские установки. С помощью этих данных можно отобразить:

- страницы, представляющие наибольший интерес для пользователя;
- новые рекламные предложения и новые версии, которые могут заинтересовать пользователя.

Например, клиент впервые посещает сайт компании "FiveLakes" и просматривает книги, относящиеся к компьютерным сетям. Вы можете сохранить эту информацию и отобразить список последних книг по данной тематике сразу же после регистрации этого пользователя на сайте. Можно также создать ссылки на какую-либо рекламную информацию об этих книгах, предоставленную издательствами.

Аналогичным образом вы можете отобразить подробную информацию о бизнес-партнерах, которые в настоящий момент зарегистрировались на сайте компании "Ramona", в программе Site Manager.

Вы способны отправить пользователю персонифицированные информационные бюллетени и сообщения электронной почты, последние вы можете также отослать тем пользователям, которые не посещали сайт свыше определенного периода времени. Например, вы можете отправить электронную почту определенным категориям пользователей, просматривающим новые книги, представленные в настоящий момент в онлайновом магазине. Кроме того, имеется возможность отправки "напоминающих" сообщений тем пользователям, которые не посещали сайт длительное время.

Методы записи данных состояния сеанса

Состояние сеанса носит временный характер. Например, если пользователь зарегистрировался при работе с ASP-приложением, то по истечении двадцати минут непрерывного простоя сеанс прекращается, а данные о состоянии сеанса утрачиваются. Вы можете указать такой допустимый период простоя, который будет превышать заданное по умолчанию значение, равное 20 минутам. Как только пользователь вновь открывает браузер, запрашивает другую страницу или же выполняет некоторую задачу после периода простоя, тогда начинается новый сеанс.

Существует несколько способов сохранения данных сессии:

- использование cookies (специальных маркеров Internet);
- применение объекта `Session`, поддержка которого реализована в ASP-страницах и Internet Information Server (IIS);
- работа с AUO и Membership Directory.

Рассмотрим подробно каждый из этих методов.

Cookies

Cookie (или cookie-файл) представляет собой маленький файл, предназначенный для идентификации пользователей, который сохраняется Web-сервером в браузере пользователя. Он может быть сохранен на жестком диске или же содержится только в памяти браузера — в зависимости от срока его действия. Всякий раз, когда пользователь запрашивает ASP-страницу, этот маркер отсылается на Web-сервер. Cookies могут хранить информацию о пользователях. Для записи информации в cookie и считывания используется коллекция `Cookies` объектов `Response` и `Request`.

Для того чтобы создать cookie с заданным для него именем и значением, воспользуйтесь следующим кодом:

```
<%Response.Cookies ("Emailbox")= "John"%>
```

Этот код создаст маркер cookie с именем Emailbox, имеющий значение John. Если маркер с таким именем уже существует, то значение John будет присвоено существующему маркеру.

Если вы хотите использовать маркер cookie для отслеживания последующих сеансов пользователя, то вам необходимо будет сохранить этот маркер на компьютере пользователя. Для этого надо воспользоваться атрибутом Expires объекта Response и указать в качестве его значения некую дату в будущем:

```
<%
   Response.Cookies ("EMailbox")= "John"
   Response.Cookies ("EMailbox").Expires = "December 31, 2001"
%>
```

Для выборки значения маркера cookie используется коллекция Request.Cookies. Например, следующий код извлечет значение John:

```
<%=Request.Cookie ("EMailbox") %>
```

Cookie хранит путь к приложению, содержащему ASP-страницу, в которой инициируется создание этого маркера. Например, ASP-файл, который создал cookie с именем Emailbox, является частью приложения Mailapp. Следовательно, cookie с именем Emailbox содержит данные вида "путь приложения /Mailapp". Когда какой-либо пользователь посылает запрос на ASP-файл, входящий в приложение Mailapp, cookie, хранящий данные "путь приложения/Mailapp" пересылается на Web-сервер.

В cookie можно указать путь. Например, вы хотите указать путь /Mailapp/Userprofiles/Data в cookie с именем Emailbox. Это можно сделать с помощью следующего кода:

```
<%Response.Cookies("Emailbox").Path = "/Mailapp//Userprofiles/
Data"%>
```

Обратите внимание на то, что здесь для указания пути использовался атрибут Path коллекции Response.Cookies collection.

Ограничение в использовании cookies заключается в том, что в настройках браузера на клиента пользователя работа с cookies может быть заблокирована. В этом случае вам потребуется применить альтернативный метод отслеживания информации о пользователе. Одним из таких методов является использование объекта Commerce.ShopperManager.

Объект *Session*

Другим распространенным методом отслеживания информации о пользователе является применение объекта Session и уникального идентификатора сеанса SessionID, вырабатываемого ASP-сервером.

Идентификатор SessionID создается с началом сеанса пользователя и однозначно идентифицирует данный сеанс. Сеанс начинается с запроса ASP-страницы в приложении.

SessionID хранится в качестве cookie браузером пользователя и применяется в течение всего сеанса. Когда пользователь запрашивает другую страницу в приложении, активная серверная страница для выполнения данного запроса проверяет заголовок HTTP-запроса на соответствие этому cookie. Поскольку данный cookie запоминается браузером пользователя, точно такой же cookie используется для сеанса в целом. Однако если вы перезапустите сервер, или же пользователь закроет браузер, то данный cookie, содержащий SessionID, будет удален из оперативной памяти и следующая сессия начнется с другим cookie, содержащим новый идентификатор SessionID.

Объект Session создается сервером IIS, когда начинается сеанс, и хранится в оперативной памяти Web-сервера. Этот объект удаляется из памяти по завершении или отмене сеанса.

Примечание

Объект Session может использоваться для хранения информации о пользователе лишь в том случае, если в браузере разрешена работа с cookies.

Объект Session является встроенным объектом ASP, предоставляющим информацию о пользователе для определенного сеанса. Информация о пользователе хранится в качестве значений переменных в объекте Session. Когда пользователь переходит от одной страницы к другой в приложении или же вызывает ASP-файл из другого приложения, значения этих переменных сохраняются.

Для того чтобы сохранить переменную в объекте Session, необходимо в нем создать поименованное вхождение (элемент) и присвоить ему некоторое значение.

Например, вы создаете вхождение First Name и присваиваете ему некоторое имя, воспользовавшись для этого следующим кодом:

```
<%
   Session ("First Name") = "Jon"
%>
```

Чтобы получить значение, хранящееся в объекте Session, необходимо указать соответствующий поименованный элемент. Например, вы можете отобразить имя зарегистрировавшегося пользователя с помощью следующего выражения:

```
Welcome <% = Session ("First Name") %>
```

Объект Session можно использовать для запоминания значений параметров, установленных пользователем, например, для сохранения значения разре-

шения экрана или установленного режима отображения страниц — текстового или с возможностью отображения графики. Эти установки можно сохранить на первой странице приложения и использовать их для всех последующих страниц. Такой подход позволяет учитывать индивидуальные предпочтения пользователей при отображении информации сайта.

Хотя использование объекта `Session` представляет собой быстрый и простой способ хранения данных о состоянии сеанса, оно ограничено рамками одного Web-сервера. Информация о сеансе работы ASP храниться на одном Web-сервере, и все страницы должны быть расположены на том же сервере, для того чтобы у них был доступ к этой информации.

Если используется несколько серверов (server farm), то URL-запросы к сайту перенаправляются на любой из них, оказавшийся в данный момент свободным. Для обеспечения работы с данными о состоянии сеанса вам необходимо убедиться в том, что все HTTP-запросы в данном сеансе перенаправлены на этот же сервер. Это можно сделать, написав соответствующий программный код для процедуры `Session_OnStart`.

Такого рода проблемы могут быть разрешены при помощи AUO и Membership Directory, реализованных в Site Server.

Сохранение данных о сеансе с помощью AUO и Membership Directory

В следующих разделах вы можете изучить:

- роль, которую играют на сайте так называемые службы принадлежности (Membership Services);
- компоненты Membership Services;
- применение AUO на сайте;
- конфигурирование AUO;
- создание экземпляра сервера принадлежности (Membership-сервера).

Службы принадлежности Membership Services

Site Server позволяет хранить данные о пользователе в базе данных и обращаться к ним в ASP-страницах.

Вы можете хранить данные о конкретном сеансе и пользователе в Membership Directory (каталоге принадлежности) и получать доступ к ним с помощью AUO. Прежде чем приступить к изучению этого подхода, мы сначала вкратце рассмотрим службы принадлежности (Membership services), предоставляемые Microsoft Site Server.

Membership Services позволяют хранить данные о пользователе и работать с ними. Вы можете создать наборы пользовательских параметров, провести идентификацию пользователя, и управлять доступом к содержимому своего сайта. Это содержимое может быть отображено с учетом предпочтений конкретного пользователя. Для реализации указанных возможностей в Site Server поддерживается Membership Directory и Membership Server.

Компоненты Membership Directory

Membership Directory представляет собой центральный репозитарий (repository) данных, в котором хранятся пользовательские данные и адресная книга. В нем могут также содержаться данные об организации, указывающие на различные информационные источники, а также любая другая информация, которая способствует отслеживанию предпочтений пользователя.

Membership Directory включает в себя:

- базу данных каталога принадлежности (Membership Directory database);
- дерево информации каталога (DIT, Directory Information Tree);
- динамические данные (Dynamic Data).

База данных Membership Directory

В качестве базы данных Membership Directory может служить база данных Microsoft Access или Microsoft SQL Server, включающая в себя данные о пользователе, данные о сайте и так называемую схему (schema).

Данные о пользователе (user data) наборы пользовательских характеристик (профилей), имя и пароль пользователя (security credentials) и группы безопасности (security groups) для данного сайта. Вы можете собирать эти данные, используя либо регистрационные формы (анкеты), в которые пользователи будут вводить необходимую информацию, либо запуская соответствующий скрипт, осуществляющий сбор значений пользовательских установок.

Данные о сайте (site data) представляют собой информацию о приложении, работающем с Membership Directory, списки распространения, содержащие адреса электронной почты тех пользователей, которым вы будете отправлять информацию, а также *словарь сайта* (site vocabulary). Этот словарь является некоторой иерархической структурой терминов, служащей для классификации содержимого документов и интересов пользователей.

Схема (schema) определяет синтаксис для всех объектов в Membership Directory. Она хранится в виде ряда объектов в базе данных Membership Directory. Существует два типа объектов-схем:

- объекты-схемы классов (class schema objects), определяющие объекты, которые должны храниться в Membership Directory, а также атрибуты, которые необходимо использовать для каждого объекта класса;

- объекты-схемы атрибутов (attribute schema objects), которые определяют отдельные атрибуты. Эти атрибуты могут быть назначены объектам класса.

Дерево объектов Directory Information Tree

Вся информация в Membership Directory представлена в виде некоторой иерархической структуры — DIT. DIT является деревом объектов, заключающем в себе объекты-контейнеры (container objects) и объекты-листья (leaf objects).

Объекты-контейнеры содержат дочерние объекты, которые, в свою очередь, могут быть контейнерами или листьями.

Объекты-листья не имеют дочерних объектов, а представляют отдельные элементы в Membership Directory. Поэтому в DIT имеются объекты-листья для представления данных о пользователе (user data), данных о группе (group data), схеме, указателей на источники информации, а также для конфигурации базы данных.

В DIT элементы называют организационными модулями (единицами). Самые нижние узлы (или организационные модули) обозначаются при помощи терма `ou=`. Организационный модуль представляет собой контейнер, в котором могут содержаться другие организационные модули.

Корневой узел в DIT представляет некую организацию и обозначается с помощью терма `o=`. Для указания наименования организации вы задаете значение для `DirectoryName` в процессе создания Membership Directory. Позднее вы не можете изменить это значение.

По умолчанию DIT имеет следующую организацию.

- Организационный модуль Admin (администратор), содержащий объекты-схемы (schema objects), объекты-источники информации (content source objects), а также словарь сайта (site vocabulary). Объекты-схемы определяют структуру данных о пользователе в Membership Directory, а объекты-источники информации указывают источники индексированного содержимого сайта. Словарь сайта определяет термы, которые могут применяться для классификации других объектов.

- Организационный модуль Members (члены), содержащий объекты-пользователи (user objects), которые представляют пользователей сайта. Эти пользователи подразделяются на зарегистрированных пользователей (registered users) и тех, информация о которых поступает из cookies (cookie users). В данном контейнере имеется субконтейнер (sub-container), `ou=Anonymousers`, предназначенный для хранения данных cookies.

- Организационный модуль Groups (группы), содержащий объекты-группы, предназначенные для представления групп, созданных с целью обеспечения администрирования и безопасности. Любая созданная вами

группа помещается непосредственно в этот контейнер. У него есть субконтейнер, `ou=NTGroups`, предназначенный для хранения групп, которые непосредственно преобразуются в группы Microsoft Windows NT groups.

- Организационный модуль Dynamic (динамические данные), представляющий непостоянные данные о пользователях, соединенных в текущий момент с сайтом. Объекты в этом контейнере созданы с помощью клиентов Microsoft NetMeeting и представляют пользователей, в текущий момент времени зарегистрированных на данном сайте. Эти данные не записываются на диск.
- Организационный модуль Applications (приложения), содержащий информацию для приложений (например, для NetMeeting).
- Организационный модуль DistributionLists (списки распространения), хранящий адреса электронной почты для отправки различной информации, например информационных или рекламных бюллетеней.

Динамические данные

Динамические данные указывают на информацию о сеансах и пользователях, зарегистрированных в текущий момент времени на сайте. Динамические данные включают в себя идентификаторы пользователей и их IP-адреса. Они также содержат информацию, относящуюся к конкретному приложению, например перечень страниц, которые посещал тот или иной пользователь, или же список товаров, помещенных в покупательскую корзину.

Динамические данные существуют в течение сеанса. В начале сеанса создается элемент с заданным значением времени жизни (TTL, time-to-live). Когда значение этого элемента достигает нуля, он удаляется из каталога принадлежности. Это гарантирует, что при разрыве или завершении сеансов данные об их состоянии не будут накапливаться.

Membership Server

Помимо каталога принадлежности, Site Server также обеспечивает работу Membership Server, реализующего сервисы (службы) принадлежности (Membership Services). Membership Server позволяет серверу приложения, такому как, например, Internet Information Server (IIS), реализовывать эти функциональные возможности на сайте. Membership Server предоставляет службы, помогающие работать с каталогом принадлежности. Вы можете также осуществлять аутентификацию и авторизацию тех пользователей, учетные данные о которых хранятся в этом каталоге.

Membership Server включает в себя следующие компоненты:

- Authentication Service (служба аутентификации);
- Active User Object (AUO) (объект активного пользователя);

- Light-weight Directory Access Protocol (LDAP) Service (служба облегченного протокола доступа к сетевому каталогу);
- Site Server Message Builder Service (служба компоновки сообщений Site Server).

Рассмотрим подробно каждый из этих компонентов.

Служба Authentication Service

Служба аутентификации реализует безопасность сайта. Для этого данный сервис выполняет следующие задачи:

- осуществление проверки достоверности паролей, указанных пользователем, путем сравнения с паролями, хранящимися в Membership Directory;
- создание контекстов безопасности (security contexts) для поддержки управления доступом к данным и приложениям в Windows NT;
- преобразование групп, заданных в группы, хранящиеся в базе данных каталога Windows NT Server.

Сервис аутентификации также позволяет извлекать свойства пользователей (user properties) из Membership Directory и осуществлять их буферизацию в целях повышения производительности ASP-приложений.

Объект активного пользователя Active User Object

AUO представляет набор атрибутов пользователя во всех источниках данных. Этот объект может интегрировать данные из различных каталогов принадлежности внешних баз данных, а также создавать виртуальные схемы, которые можно использовать в скриптах или приложениях.

С помощью AUO, приложение получает доступ к свойствам пользователя, работающего в текущий момент времени, не прибегая при этом к программированию процессов идентификации пользователя, указания местоположения этих данных. AUO может осуществлять считывание/запись из источника данных, а также создавать объекты. Аналогичные функции могут выполнять скрипты и программы, использующие AUO.

Служба Light-weight Directory Access Protocol Service

Эта служба предоставляет интерфейс для независимого от платформы доступа к каталогу принадлежности. Причем доступ обеспечивается как по отношению к постоянным, так и к динамическим данным в этом каталоге. С помощью этого сервиса вы можете извлекать, редактировать, включать и удалять различные записи (entries) базы данных каталога принадлежности.

Вы можете установить и настроить службу LDAP на том же компьютере, где расположена база данных каталога принадлежности, на компьютере с сервером приложения или же на отдельном компьютере.

Для осуществления доступа к каталогу принадлежности авторизованное приложение, поддерживающее LDAP, может подключиться к сервису LDAP. Сервис аутентификации подключается к сервису LDAP для получения информации и обеспечению доступа к содержимому каталога принадлежности.

Служба Site Server Message Builder Service

Эта служба позволяет отправлять электронную почту с помощью Direct Mail тем адресатам, которые указаны в списке распространения каталога принадлежности. Direct Mail представляет собой средство персонализации, предоставляемое в Microsoft Site Server.

Обсуждение Direct Mail выходит за рамки данной книги. Полное рассмотрение List Builder, Message Builder и Direct Mail содержится в документации Site Server.

Теперь давайте рассмотрим, как используется AUO в ASP-приложении для хранения данных о состоянии сеанса.

Применение AUO на сайте

Для получения доступа к данным о состоянии сеанса с помощью AUO вам необходимо настроить AUO на сайте. Для этого выполните следующие шаги:

1. Создайте базу данных для Membership Directory.
2. Создайте экземпляр Membership Server с AUO на требуемом Web-сервере.
3. С помощью Membership Server отобразите сайт, данные о сеансе работы с которым необходимо сохранить.
4. Настройте AUO в Membership Server.

Создание базы данных SQL Server для Membership Directory

Для создания базы данных каталога принадлежности вы можете воспользоваться Microsoft Access или Microsoft SQL Server. MS Access можно использовать в том случае, если база данных предназначена для малого сайта, насчитывающего несколько пользователей. Максимальный объем базы данных MS Access составляет 1 Гбайт.

Для сайтов, обслуживающих несколько сотен зарегистрированных пользователей, следует выбрать базу данных Microsoft SQL Server.

Примечание

Вы не можете перенести базу данных каталога принадлежности, созданную в MS Access, на SQL Server. Если вам захочется перейти на использование SQL Server, то придется создавать базу с самого начала.

Создание экземпляра Membership Server

Для установки Membership Server необходимо следующее программное обеспечение:

- Microsoft Windows NT Server 4.0;
- Windows NT 4.0 Service Pack 3;
- Windows NT file system (NTFS);
- Microsoft Internet Explorer 4.01 или выше;
- Windows NT 4.0 Option Pack;
- Microsoft Site Server 3.0.

Когда вы выполняете выборочную инсталляцию Membership Server и выбираете частные опции Personalization и Membership из общей опции Knowledge, то автоматически создаются Membership Server и Membership Directory. Этот сервер называется Intranet (Windows NT Authentication) Membership Server. Он содержит следующие компоненты:

- AUO;
- LDAP service;
- Message Builder Service.

Однако в этом каталоге принадлежности применяется база данных MS Access. Для того чтобы использовать для него базу данных SQL Server, вам потребуется создать новый экземпляр Membership Server и новый каталог принадлежности.

Перенос Web-сервера на Membership Server

Для использования AUO на Web-сервере или Web-сайте для хранения данных о сеансе вам необходимо перенести данный сервер в экземпляр Membership Server. Вы можете перенести один и более серверов в один и тот же экземпляр Membership Server, служащий для хранения и выборки данных. Однако для каждого сайта может использоваться лишь один Membership Server.

Настройка AUO на Membership Server

Как уже обсуждалось ранее, AUO осуществляет доступ к различным источникам данных и представляет собой виртуальную схему, доступную для приложений. Это позволяет приложениям взаимодействовать с единым интерфейсом данных о пользователе.

Примечание

В большинстве случаев Microsoft рекомендует, чтобы AUO использовался лишь для доступа к Membership Directory.

Для того чтобы настроить AUO для работы на сайте, необходимо:

- сконфигурировать AUO-провайдер (AUO-провайдером является источник данных для AUO);
- создать или выбрать класс объектов для хранения данных о сеансе. Для этого класса должна быть предусмотрена поддержка в каталоге принадлежности. Например, вы можете использовать класс Members, предоставляемый в Site Server.

Конфигурирование AUO-провайдера

Вы можете задать один и более источников данных для AUO. В качестве AUO-провайдеров можно указывать следующие источники данных.

- *Membership Directory*. В качестве провайдера можно указать контейнеры, содержащиеся в каталоге принадлежности. Корневым провайдером является содержащийся в этом каталоге контейнер Members.
- *Directory Services (службы каталогов)*. Эти службы должны быть совместимы с LDAP, Novell Directory Services (NDS), Netware 3.0 (NWCOMP), IIS или с Windows NT Active Directory Services.
- *Базы данных, предусматривающие поддержку ODBC*. Эти базы данных не заданы в качестве каталога принадлежности, однако они могут быть настроены для хранения данных о пользователях.
- *Базы данных без поддержки ODBC*. Для этих баз данных необходим специальный ADS-провайдер.

Для каждого AUO-провайдера вам необходимо установить следующие опции.

- Provider Alias (алиас провайдера), который уникальным образом идентифицирует AUO-провайдера. Вы можете использовать для него лишь алфавитно-цифровые символы.
- Путь ADS для контейнера данных, указывающих тип провайдера, его местоположение и контейнер в дереве информации каталога (Directory Information Tree). Этот контейнер содержит данные, поступившие от этого провайдера. Путь ADS для Membership Directory имеет следующий вид: LDAP://<*имя_сервера:порт*>/o=<*имя_каталога*>/ou=members.
- Путь ADS к объекту класса Schema для создания новых объектов данных для провайдера. Относительно Membership Directory, этот объект будет размещаться в контейнере Schema дерева информации каталога (DIT), и путь будет иметь следующий вид: LDAP://<*имя_сервера:порт*> /o=<*имя_каталога*>/ou=admin/cn =schema/cn=member.
- Суффикс пути ADS, указывающий объект, с которым будет работать AUO. Если вы укажете в качестве суффикса имя пользователя, то AUO

будет осуществлять поиск информации об этом пользователе, представленной в виде o=<*имя_каталога*>/ou=members/cn=user name.

Кроме конфигурирования AUO-провайдера, вам также необходимо создать класс объектов для хранения данных. Для этого надо сконфигурировать новый контейнер, новый класс и атрибуты. Все это понадобится для приема данных о состоянии сеанса на ваших ASP-страницах.

Каталог принадлежности — Membership Directory — содержит объекты-схемы, определяющие классы и атрибуты для каждого содержащегося в нем объекта. Каждый объект представляет собой экземпляр класса и состоит из атрибутов, за которыми следует соответствующий класс.

Рассмотрим пример конфигурирования AUO для сайта FiveLakes. Для этого вам нужно выполнить перечисленные ниже шаги.

1. Создайте контейнер с именем `ou=SessionStateData`, расположенный под корневым узлом каталога принадлежности.
2. Создайте в схеме атрибут с именем `pageview` типа `Integer`.
3. Создайте объект `Schema` схемы с именем `SessionState`. В этом объекте должны быть указаны атрибуты `cn` и `objectclass`.
4. Укажите организационный модуль в качестве родительского класса для того, чтобы можно было создать экземпляр класса `SessionState`.
5. Укажите в качестве атрибута `cn` — `RDNAttribute`.

Вам нужно создать вспомогательный AUO-провайдер для данного сайта с именем FiveLakes. Путь ADS для данного провайдера будет иметь следующий вид: <*имя_компьютера:порт_LDAP*>/o=<*имя_каталога*>/ou=SessionStateData.

Путь схемы для AUO-провайдера имеет следующий вид: <*имя_компьютера:порт_LDAP*>/o=<*имя_каталога*>/ou=admin/cn=schema/cn= SessionState.

Работа с данными о состоянии сеанса с помощью AUO

В следующих разделах вы можете изучить вопросы:

- как отредактировать ASP-страницу, чтобы организовать доступ к данным о состоянии сеанса, или их хранение;
- как осуществить поиск данных о пользователе в каталоге принадлежности.

Модификация ASP-страницы

Для хранения этих данных и работы с ними вам необходимо создать экземпляр AUO. В ту ASP-страницу, с помощью которой вы собираетесь хранить данные о состоянии сеанса, вам нужно будет внести определенный про-

граммный код. Например, вы можете включить код в страницу default.asp приложения, чтобы сохранить данные о сеансе для этой страницы, к которой пользователь имеет доступ по умолчанию.

Если же вы хотите иметь данные о сеансе для всех страниц, к которым обращался пользователь, то вы можете создать маленький ASP-файл, в котором будет содержаться код, предназначенный для обеспечения хранения данных о состоянии сеанса в каталоге принадлежности. В этом коде будет использоваться AUO. Затем вам следует включить этот файл во все страницы приложения.

Рассмотрим код для хранения данных сеанса в странице default.asp сайта FiveLakes. Чтобы сформировать экземпляр AUO, вам нужно создать экземпляр объекта `UserObjects`:

```
Set user = Server.CreateObject("Membership.UserObjects")
```

Этот фрагмент кода порождает AUO и присваивает ему имя пользователя, подключенного в текущий момент к данной странице. Теперь вы можете получить статические характеристики (свойства) пользователя (данные статического AUO-провайдера).

Для того чтобы получить динамические данные о сеансе, вам необходим динамический AUO-провайдер FiveLakes. Нужно также создать объект сеанса (session object) для хранения данных о сеансе и доступа к ним. Для сайта FiveLakes имя AUO-провайдера данных о состоянии сеанса имеет значение `FiveLakesAUO`, а имя класса для объекта сеанса — `SessionState`.

Сначала вы запишите код, который осуществляет проверку того, что объект сеанса уже создан. Если такого объекта нет, то вы его создаете:

```
If not IsArray(user("SessionState".objectclass) Then
   User("SessionState").objectclass = Array("SessionState", _
   "dynamicobject")
end If
```

Для назначения продолжительности сеанса укажите значение времени жизни (TTL) данного объекта.

```
User("SessionState").entryTTL = 600
```

Создав объект сеанса, вы можете указать значения его атрибутов и получить доступ к его свойствам точно так же, как и для любого другого объекта. Например, для того чтобы сохранить имя страницы, к которой обращался некий пользователь, в атрибуте `pagename` вы можете написать следующий код:

```
User("SessionState").pagename = Request("SCRIPT_NAME")
```

Чтобы извлечь имя страницы, к которой обращался пользователь, примените следующий код:

```
pageAccessed = User("SessionState").pagename
```

Далее вы можете зафиксировать эти изменения с помощью метода `setInfo`:

```
user.setInfo
```

Поиск данных о состоянии сеанса в Membership Directory

С помощью AUO можно осуществлять поиск как постоянных, так и динамических данных о свойствах сеанса или пользователя в Membership Directory. Например, вы можете сгенерировать список всех пользователей, зарегистрированных в текущий момент времени на сайте, или же тех, кто в настоящий момент времени просматривает какую-либо страницу.

На сайте FiveLakes имя страницы, к которой производится обращение в текущий момент времени, хранится в качестве значения свойства `pagename`. Вы можете осуществить запрос к Membership Directory с помощью ADO (Active Data Objects) и сгенерировать список всех пользователей обратившихся к данной странице.

```
Set conMD = CreateObject("ADODB.Connection")
ConMD.Provider = "ADsDSOObject"
ConMD.Open "Ads Provider"
Querystring = "<LDAP://INSTRUCTOR:1003/o=FiveLakes/
ou=SessionStateData/>; (&(objectClass = SessionState)(currentPath="
& Request("SCRIPT_NAME") &"));ADsPath,cn;SubTree"
Set rs = conn.Execute( queryString, numRecords, 1)
Do While Not rs.EOF
   Response.Write rs("cn") & "<br>"
   rs.MoveNext
Loop
```

Преимущества AUO и Membership Directory для сохранения данных о состоянии сеанса

Данные о состоянии сеанса хранятся в виде динамических данных в Membership Directory. Вы можете сконфигурировать AUO для создания динамического объекта для каждого пользователя в Membership Directory. Атрибуты этого объекта являются атрибутами сеанса. Динамический объект в Membership Directory имеет время жизни (TTL), значение которого задается при создании объекта. Как только значение TTL истекает, динамический объект удаляется из Membership Directory. Это происходит в тех случаях, когда пользователь не обращается к сайту на протяжении времени, превышающего значение TTL.

Если пользователь посещает сайт во время существования динамического объекта, то этот объект обновляется и начинается новый сеанс.

Преимущества использования динамического объекта заключаются в том, что вы можете хранить данные о состоянии сеанса работы пользователя на Web-сайте, реализованном на множестве серверов (Web farm). При этом пользователь может обращаться к страницам, хранящимся на различных серверах. Однако для такого сайта существует лишь один каталог принадлежности, и динамический объект хранится в этом каталоге. Это гарантирует, что в течение одного сеанса будет создан один динамический объект.

Кроме того, с данными о пользователе можно работать даже в том случае, если его запрос направлен на другой сервер. Смысл заключается в том, что все страницы получают доступ к данным динамического объекта в Membership Directory и осуществляют запись о состоянии сеанса только в этот динамический объект.

Использование AUO более устойчиво к ошибкам, чем применение объекта `Session`. Если вы используете объект `Session` для сохранения данных о сеансе и в это время сервер "разваливается", то данные будут утеряны. Однако в подобной ситуации при использовании AUO для сохранения данных о состоянии сеанса в Membership Directory данные будут сохранены.

Лабораторная работа 10. Сохранение данных о состоянии сеанса

Цели

После выполнения этой лабораторной работы вы должны уметь:

- создавать Membership Server и переносить его на Web-сервер;
- конфигурировать AUO для Membership Server.

Перед началом работы

Предварительные требования

Перед началом работы необходимо:

- иметь основные понятия об HTML и ASP-скриптах;
- быть знакомым с Microsoft Internet Explorer и Microsoft Visual InterDev 6.0.

Подготовка к лабораторной работе

Для выполнения данной работы не требуется каких-либо дополнительных действий.

Примечание

Время, необходимое для выполнения данной работы, — 45 минут.

Упражнение 1. Установка Membership Server

В этом упражнении мы создадим Membership Server для сайта Ramona. В качестве каталога принадлежности на этом сервере будет использоваться SQL Server RamonaMD.

Для создания базы данных SQL Server с именем `RamonaMD` выполните перечисленные ниже действия:

1. В меню кнопки **Start** укажите пункт **Programs**, а затем последовательно команды **Microsoft SQL Server 7.0** и **Enterprise Manager**.
2. В левой панели щелкните мышью на знаке +, расположенном после **SQL Server Group**.
3. В левой панели щелкните мышью на знаке +, расположенном после *<local server>*.
4. В левой панели щелкните правой кнопкой мыши по элементу **Databases**, а затем выберите команду **New Database**.
5. В поле **Name** диалогового окна **New Database** введите `RamonaMD` и нажмите кнопку **OK**.
6. В левой панели щелкните по элементу **Databases**.
7. В правой панели щелкните правой кнопкой мыши по элементу **RamonaMD**, а затем выберите команду **Properties**.
8. В диалоговом окне **RamonaMD Properties** щелкните мышью по вкладке **Options**.
9. Выберите флажок **Truncate log on checkpoint**, а затем нажмите кнопку **OK**.
10. Закройте Enterprise Manager.

Для запуска MMC[1] необходимо в меню кнопки **Start** указать пункт **Programs**, а затем последовательно выбрать команды **Microsoft Site Server**, **Administration**, **Site Server Service Admin (MMC)**.

Для создания экземпляра Membership Server:

1. В левой панели щелкните мышью на знаке +, расположенном после элемента **Personalization and Membership** (Персонализация и принадлежность).
2. Щелкните правой кнопкой мыши *<localhost>*, выберите команду **New**, а затем щелкните по элементу **Membership Server Instance** (Экземпляр Membership Server).
3. Произойдет запуск New Membership Server Wizard (Мастера создания нового сервера принадлежности). Нажмите кнопку **Next**.

---

[1] MMC — Microsoft Management Console (консоль администрирования). — *Пер.*

4. Создайте Membership Server для хранения данных о состоянии сеанса, выбрав опцию **Custom configuration** (Выборочная конфигурация), а затем нажмите кнопку **Next** в окне **Select Configuration Mode** (Выбор режима конфигурации).
5. В окне **Select Configuration Options** (Выбор опций настройки) отметьте инсталлируемые компоненты. Установите флажки **Active User Object (AUO)** и **LDAP Service**, а затем щелкните по кнопке **Next**.
6. В окне **Select the Membership Directory** (Выбор каталога принадлежности) укажите, что вы хотите создать новый каталог принадлежности (Membership Directory) для Membership Server, а затем щелкните по кнопке **Next**.
7. В окне **Select the Authentication Mode** (Выбор режима аутентификации) оставьте опцию **Membership Authentication**, принятую по умолчанию, а затем нажмите кнопку **Next**.
8. В окне **Name the Membership Directory and Create Account** (Присвоение имени каталогу принадлежности и создание учетной записи) в поле **Membership Directory name** (Имя каталога принадлежности) введите `RamonaMD`.
9. Введите `ramonaadmin` в поле **Password** (Пароль). Подтвердите пароль, а затем щелкните по кнопке **Next**.
10. В окне **Select the Database Type** (Выбор типа базы данных) отметьте опцию **Microsoft SQL Database**, а затем нажмите кнопку **Next**.
11. В окне **Type SQL Database Information** (Ввод информации о базе данных SQL) укажите местоположение базы данных. В поле **Server name** (Имя сервера) введите `localhost`, а в поле **Database name** (Имя базы данных) — `RamonaMD`.
12. Введите `sa` в поле **SQL Server database user name** (Имя пользователя базы данных SQL Server), а затем нажмите кнопку **Next**.
13. В окне **Create Local LDAP Service** (Создание локального сервиса LDAP) вам необходимо указать IP-адрес и номер порта для сервиса LDAP. Примите значения, заданные по умолчанию, а затем нажмите кнопку **Next**.
14. Нажмите кнопку **Finish**.

Membership Server с именем **Membership Server #2** создан.

Для того чтобы перевести Web-сервер на новый Membership Server:

1. В меню кнопки **Start** укажите **Programs**, далее **Microsoft Site Server**, **Administration**, а затем выберите команду **Site Server Service Admin (HTML)**.
2. В окне **Web Administration** щелкните **Personalization and Membership**.
3. В левом фрейме щелкните **Web Site Mappings**.

4. В окне **Web Site Mappings for <*localhost*>** выберите **Default Web Site**.
5. В окне списка **Map to** выберите **Membership Server #2**, а затем нажмите кнопку **Submit**.

Упражнение 2. Конфигурирование AUO для сайта

Предположим, вы хотите сохранить информацию о заказе на сайте Ramona. Для каждого из них эта информация включает в себя данные заказа, категорию заказанных книг и их количество.

В этом упражнении вы создадите и настроите на сайте Ramona объект AUO, который для каждого сеанса будет запоминать всю необходимую информацию.

Для того чтобы создать контейнер для хранения изменений в заказе:

1. В левой панели консоли Site Server щелкните правой кнопкой мыши на элементе **Membership Directory Manager**, укажите команду **New**, а затем команду **Container**.
2. Появится диалоговое окно **New Container Wizard**. Щелкните по кнопке **Next**.
3. Введите `orderbehavior` в качестве имени данного контейнера, а затем нажмите кнопку **Finish**.

Чтобы создать атрибуты для AUO :

1. В левой панели дважды щелкните мышью на **Membership Directory Manager**.
2. В левой панели дважды щелкните **ou=Admin**.
3. В левой панели дважды щелкните мышью **cn=Schema**, укажите **New**, а затем щелкните **Attribute**.
4. Произойдет запуск New Attribute Wizard (Мастера создания нового атрибута). Нажмите кнопку **Next**.
5. В окне **Type the Name of the New Attribute** (Ввод имени нового атрибута) введите `dateoforder` в поле **Name** и `Date of Order` в поле **Display name**, а затем нажмите кнопку **Next**.
6. В окне **Select the Attribute Syntax** (Выбор синтаксиса атрибута) укажите **Date/Time**, а затем щелкните по кнопке **Finish**.
7. Повторите шаги 3—6 для создания атрибута категории книги (`bookcategory`) и атрибута количества (`orderquantity`). Для атрибута `bookcategory` укажите значение синтаксиса — **String**, а для `orderquantity` — **Integer**.

Для создания класса purchaseorder:

1. В левой панели дважды щелкните правой кнопкой на элементе **cn=Schema**, укажите **New**, а затем щелкните **Class**.
2. Произойдет запуск New Class Wizard (Мастера создания нового класса). Нажмите кнопку **Next**.
3. В окне **Type the Name of the New Class** (Ввод имени нового класса) укажите purchaseorder в качестве имени класса, а затем щелкните по кнопке **Next**.
4. В окне **Add Parent Classes** (Добавить родительские классы) щелкните по кнопке **Remove** для удаления организации как одного из родителей класса, а затем нажмите кнопку **Next**.
5. В окне **Select the Attributes for the Class** нажмите кнопку **Add**.
6. В диалоговом окне **Add Attribute** выберите атрибут **Date of Order attribute**, а затем щелкните по кнопке **OK**.
7. Повторите шаги 5 и 6 для того, чтобы включить атрибуты bookcategory и orderquantity в данный класс.
8. Отметьте указатель, расположенный после атрибута bookcategory.
9. Выберите bookcategory в окне списка **Naming attribute**, а затем нажмите кнопку **Finish**.

Для того чтобы настроить дополнительный/вторичный AUO-провайдер для сервера **Membership Server #2**:

1. В левой панели щелкните правой кнопкой мыши **Membership Server #2**, а затем выберите команду **Properties**.
2. В диалоговом окне **Properties** сервера **Membership Server #2** выберите вкладку **Active User Object (AUO) Providers**.
3. Нажмите кнопку **Add**.
4. Появится диалоговое окно **Active User Object (AUO) Provider Properties**. В указанные поля введите следующую информацию:

   AUO alias: RamonaAUO

   Build Activer Directory Services (ADS) Path:

   <localhost>:LDAPportnumber/o=RamonaMD/ou=members

   (где значением по умолчанию для LDAPportnumber является 1003)

   Schema path:

   <localhost>:LDAPportnumber/o=RamonaMD/ou=Admin/cn=Schema /cn=purchaseorder

5. Нажмите кнопку **OK**.
6. Еще раз нажмите кнопку **OK**.

Вопросы для повторения

1. Когда в ASP-приложении создается сеанс?
2. Как данные о сеансе хранятся в Membership Directory?
3. Как AUO используется для работы с данными о пользователе?

ГЛАВА 11

Обработка данных кредитных карт

Краткий обзор

Одним из наиболее важных компонентов любого сайта электронной коммерции является его система обработки платежей. Для достижения максимального трафика и уровня продаж вы должны обеспечить возможность обработки данных кредитных карт на своем сайте. В этой главе вы узнаете, как осуществить прием на сайте кредитных карт в качестве платежного средства. Вы также ознакомитесь с применением Microsoft Passport для того, чтобы пользователи и бизнес-партнеры могли быстро и надежно предоставлять информацию о платежах.

После прочтения данной главы вы научитесь:

- обеспечивать возможность обработки данных кредитных карт на сайте;
- использовать Microsoft Passport.

Обеспечение обработки данных кредитных карт на сайте

В этом разделе вы познакомитесь с:

- требованиями по обработке данных кредитных карт;
- этапами обработки данных кредитных карт.

Давайте вначале рассмотрим, какие компоненты необходимы для обеспечения приема платежей по кредитным картам на сайте электронной коммерции.

Требования по обработке данных кредитных карт

Для обработки кредитных карт на сайте необходимо иметь:

- торговый счет для работы в Web (Web-enabled merchant account);
- программное обеспечение обработки платежей (payment processing software);
- безопасную форму заказа (secure order form).

Учет торговых операций на основе Web

Торговый счет (merchant account) — вид счетов, позволяющий принимать и хранить фонды (денежные средства), относящиеся к бизнес-транзакциям на сайте. Он похож на счета, которые выписываются при осуществлении почтовых заказов или розничной торговли (retail business), но для торгового счета не нужно физическое наличие кредитной карты. Когда покупатель производит покупку и оплату по кредитной карте, то денежные средства будут переданы с кредитной карты покупателя на ваш торговый счет.

Торговый счет может открыть для вас банк или какая-либо финансовая организация. Приобретение торгового счета является одним из первых шагов, который необходимо выполнить, после того как вы примите решение осуществлять бизнес с помощью Internet. В идеальном случае, следует обратиться в банк, где у вас уже существует кредит или ипотечный счет (mortgages account). Это может оказаться полезным, если вы занимаетесь бизнесом более двух лет. В зависимости от условий банка вам может понадобиться предоставить финансовые отчеты (financial statements), копии налоговых деклараций или налоговых выписок по счету (tax statements), и, возможно, сведения о сумме кредита (credit reserve).

Вам также потребуется проинформировать банк о типичном объеме заказа и предполагаемом по данному счету среднемесячном объеме продаж. В некоторых банках принято, чтобы у клиента часть денежных средств приходилась на покрытие издержек в случае мошенничества и возмещение убытков.

В качестве альтернативы непосредственному обращению в банк может служить связь с одной из независимых сервисных организаций, специализирующихся на приобретении торговых счетов для компаний. Такая организация обратится в банк от вашего имени и предоставит вам торговый счет. Вам потребуется выполнить все требования банка и оплатить сервисной организации ее услуги. Некоторые организации могут пропускать ваши транзакции через свои торговые счета. В таком случае вы будете прибегать к услугам сервисной фирмы, выполняющей все требования банка. Если данная сервисная фирма в какой-то момент больше не сможет обрабатывать

данные кредитных карт, вам придется заново обращаться за получением торгового счета, необходимого для работы в Web.

При обращении за получением торгового счета для работы в Web вам нужно указать продукцию и услуги, которые вы собираетесь продавать. Необходимо, чтобы банк их одобрил. При получении торгового счета для работы в Web вы приобретаете также торговый идентификатор (Merchant ID) и идентификатор терминала (Terminal ID).

Стоимость торгового счета для работы в Web

Торговый счет для работы в Web влечет за собой некоторые издержки. Они могут включать в себя следующие позиции:

- регистрационные сборы (application fees);
- учетные ставки (discount rates);
- отчисления за обмен (intercharge fees);
- ежемесячные отчисления (monthly fees);
- отчисления в резервный фонд или запасные затраты (reserve costs).

Когда вы обращаетесь за получением торгового счета, вам необходимо оплатить некоторые отчисления. Большинство банков взимает не подлежащий рефинансированию регистрационный сбор (non-refundable application fee), который вам не будет возмещен, если банк не одобрит вашу заявку.

Учетная ставка — это процентная величина от общей суммы продаж, взимаемая банком за проведение ваших бизнес-транзакций. Учетная ставка зависит от объема заказа, общего объема ежемесячных транзакций, а также от ваших скидок, соответствующих опыту возврата платежа.

Отчисления за обмен — это деньги, которые ваш банк будет оплачивать банку покупателя за передачу фондов со счета покупателя на ваш счет. Величина этих отчислений зависит от размера переданных фондов. Вы будете вынуждены платить банку эти отчисления за каждую транзакцию. Обычно эта плата является частью учетной ставки.

Вам также придется платить банку постоянные по величине ежемесячные отчисления за использование торгового счета. Для вас к тому же может быть установлен некий минимальный объем отчислений за транзакции, который будет взиматься, при условии, что ежемесячный объем отчислений по учетной ставке не превышает этот минимальный объем.

Наконец, вы будете вынуждены оплачивать банку любые оспариваемые расходы (contested charges). Это означает, что помимо оплаты за потерянные товары и транспортных издержек вы будете выплачивать банку возвратные платежи (charge back fees). Ваш выбор между банком и финансовой организацией будет зависеть от ежемесячной стоимости торгового счета и оценочных доходов вашей компании. Это будет также зависеть от того, какие кре-

дитные карты может обеспечивать данный банк. В идеальном случае, банк должен поддерживать все основные виды кредитных карт, таких как MasterCard, Visa, American Express и Discover.

Программное обеспечение обработки платежей

Как только у вас появится торговый счет, вам потребуется механизм, с помощью которого будет осуществляться перевод с кредитной карты покупателя на ваш торговый счет. Это выполняется с помощью программного обеспечения обработки платежей. Данное программное обеспечение взаимодействует с вашим сайтом, со счетом кредитной карты покупателя, а также с финансовой организацией, предоставившей торговый счет.

Когда покупатель делает заказ, программное обеспечение обработки платежей выполняет следующие задачи:

1. Осуществляет прием информации, введенной покупателем, о кредитной карте.
2. Устанавливает связь с акцептующей финансовой организацией, которая, в свою очередь, связывается с банком, обеспечивающим оплату по кредитной карте покупателя. Данное программное обеспечение также проверяет действительность кредитной карты.
3. Акцептующая финансовая организация выясняет, достаточно ли имеется средств на счету кредитной карты для оплаты заказанных товаров.
4. Отправляет подтверждение транзакции в банк, содержащий торговый счет.

Давайте теперь рассмотрим, какими основными возможностями должно обладать программное обеспечение обработки платежей. Оно должно быть:

- рентабельным;
- безопасным;
- легко реализуемым на Web-сайте.

Стоимость является главным фактором при выборе программного обеспечения обработки платежей. Использование данного программного обеспечения включает в себя затраты на разработку, реализацию, эксплуатацию, а также стоимость выполнения каждой транзакции.

Когда покупатель предоставляет информацию о своей кредитной карте, эта информация должна быть передана в безопасном (защищенном) режиме для проверки ее допустимости. Программное обеспечение обработки платежей проверяет допустимость этой информации, и если она подтверждается, то передает ее в ваш банк, который, в свою очередь, связывается с банком покупателя для получения согласия на перевод средств. Программное обеспе-

чение обработки платежей "отвечает" за весь процесс передачи данных, и, следовательно, должно обеспечивать их безопасность.

Примечание

О потребностях и методах обеспечения безопасности в Internet вы узнаете позже в этой главе.

Вы должны уметь быстро и с очень незначительными затратами интегрировать программное обеспечение обработки платежей в свое приложение. Время, затрачиваемое на разработку, а также усилия, необходимые для использования программного обеспечения обработки платежей должны быть незначительными, насколько это возможно.

Лучший метод для реализации обработки данных кредитных карт на вашем сайте заключается в использовании услуг одной из организаций, обеспечивающих поддержку "стандартных" (т. е. имеющихся в продаже) пакетов программного обеспечения обработки платежей.

Учитываемые факторы при выборе программного обеспечения

Вы можете избежать всех вопросов, связанных с разработкой заказного программного обеспечения, и использовать какой-либо пакет, который можно получить в онлайновом режиме. Для того чтобы можно было использовать такой пакет, вам необходимо зарегистрироваться в организации, оказывающей поддержку данного программного обеспечения. Ваш торговый банк оплатит расходы на это программное обеспечение и обработку транзакций. В различных организациях существуют разные правила оплаты обработки транзакций. В одних случаях производится оплата за каждую транзакцию, в других может требоваться оплата за некоторое количество выполненных транзакций. Например, вам может потребоваться оплатить определенную сумму за любое количество транзакций от 1 до 100, и другую сумму — за транзакции от 101 до 300.

В выборе программного обеспечения обработки платежей вам помогут следующие факторы.

- Поддержка различных типов кредитных карт.

  Программное обеспечение обработки платежей должно поддерживать все типы кредитных карт, которые вы хотите принимать на сайте.

- Отношение с вашим банком или финансовой организацией.

  Организация, осуществляющая сопровождение программного обеспечения обработки платежей и ваш банк должны тесно сотрудничать с вами, предоставляя качественное обслуживание. Многие организации по сопровождению программного обеспечения обработки платежей оказывают помощь своим клиентам в приобретении торговых счетов в банке.

- Простота реализации на сайте электронной коммерции.

  Вы должны легко интегрировать программное обеспечение обработки платежей в программное обеспечение сайта. К тому же его эксплуатация, обновление и установка новых версий должна требовать от вас минимальных усилий.

- Масштабируемость (scalability).

  Программное обеспечение обработки платежей должно справляться с растущим трафиком и объемом обработки.

- Регистрация транзакций.

  Транзакции должны регистрироваться в базе данных, к которой у вас имеется доступ. Это позволит проводить анализ транзакций и пользоваться регистрационными данными для корректировки ошибок.

- Доказательство оплаты (proof of payment).

  После перевода фондов на ваш торговый счет программное обеспечение обработки платежей должно вырабатывать доказательство оплаты, которое затем может быть отправлено покупателю.

Одними из самых популярных фирм программного обеспечения обработки платежей являются компании CyberCash, Authorize.net и PCAuthorize.

Безопасная форма заказа

Покупатели испытывают большие опасения относительно безопасности сайтов и степени доверия к ним. Существует "прирожденный" страх того, что номер кредитной карты может быть перехвачен и использован в корыстных целях. Для решения этой проблемы вам необходимо:

- гарантировать покупателю аутентичность (authenticity) вашего сайта[1];
- реализовать безопасный метод передачи секретной информации для программного обеспечения обработки платежей;
- реализовать систему, позволяющую покупателям оплачивать заказанные товары посредством безопасного интерфейса.

Гарантии аутентичности сайта для покупателей

Вы можете убедить покупателей в том, что указание информации о кредитной карте на сайте для них не представляет угрозы в силу его оформления и реализации соответствующих мер по обеспечению безопасности.

Ваш сайт должен быть разработан с учетом удобства и простоты его использования. Ниже приводится ряд рекомендаций относительно дизайна сайта.

---

[1] Аутентичность — способность устанавливать подлинность пользователя. — *Пер.*

- Для того чтобы сделать заказ, покупатели должны выполнять минимальное количество щелчков мыши.
- Кнопки требуется снабжать отчетливыми надписями. Результат выполнения каждой задачи необходимо сопровождать отчетливым уведомлением. Например, покупатель должен знать, что после того, как он щелкнул по кнопке **Order Now** (Заказать сейчас), начинается процесс обработки данных о кредитной карте.
- Подтвердив свои намерения сделать заказ, покупатель должен получить такую информацию, которая придаст ему уверенности. Вы можете отобразить страницу подтверждения заказа, содержащую номер заказа и другую, относящуюся к заказу информацию, а также контактные реквизиты (фамилия, номер телефона, e-mail) на случай возникновения проблем.
- Наименование и почтовый адрес вашей организации должен четко отображаться на экране.

В дополнение к этим мерам, вы можете вступить в одну из Internet-групп защиты интересов покупателей и поместить логотип этой группы на своем сайте. Вы можете также отразить на сайте информацию о безопасности кредитной карты и о том, как эта безопасность реализована.

Реализация мер безопасности на сайте

Рост вашей компании непосредственно зависит от того, насколько обеспечена ее защита, а также насколько безопасно чувствуют себя ваши клиенты, вводя информацию об оплате в онлайновом режиме. В качестве составной части мер безопасности вам нужно защитить сервер, аутентифицировать сайт[1] и все категории пользователей, вводящих секретную информацию на сайте. Для этого вам необходимо иметь:

- цифровой сертификат, содержащий открытый ключ;
- механизм обеспечения безопасности передачи секретных данных.

Давайте рассмотрим подробно каждое из этих требований.

Цифровой сертификат и открытый ключ

Цифровой сертификат (digital certificate) представляет собой текстовый файл, в котором содержится информация о владельце и соответствующем сертификационном центре (certification authority)[2]. В данном файле, в зашифрованном виде, содержатся также сведения для идентификации этих полномо-

---

[1] То есть снабдить его средствами установления подлинности пользователей. — *Пер.*

[2] Сертификационный центр (certification authority) — компания или организация, служащая хранилищем цифровых сертификатов. — *Пер.*

чий[1]. Цифровой сертификат используется на Web-сайте наряду с шифрованием по открытому ключу.

Шифрование по открытому ключу (public key encryption) — это криптографическая система, в которой используется два ключа — один для шифрования, а другой — для дешифрования. Это следующие ключи:

- *открытый ключ* (public key), предоставляющийся всем пользователям, которым необходимо отправлять данные на ваш сайт;
- *закрытый ключ* (private key), хранящийся у вас и установленный на сервере.

Покупатели, имеющие открытый ключ сайта, могут запросить сертификационный центр произвести проверку (верификацию) данного ключа.

Теперь рассмотрим, как цифровые сертификаты и открытые ключи позволяют обеспечить безопасную передачу данных. Рассмотрим покупателя, которому необходимо предоставить информацию о кредитной карте на сайт. Сначала он может получить цифровой сертификат и предоставить его для проверки в сертификационный центр. Если сертификат является действительным, то данные будут зашифрованы по открытому ключу и переданы на сайт. На сайте генерируется пара "открытый ключ/закрытый ключ", и открытый ключ предоставляется в сертификационный центр. Сертификационный центр добавляет открытый ключ в цифровую подпись. Когда вы принимаете данные, вы можете дешифровать их с помощью закрытого ключа. Если в процессе передачи данные будут перехвачены, их невозможно будет дешифровать, поскольку необходимый для дешифрации ключ находится только на сайте.

Для реализации этой функциональной возможности вам нужно приобрести цифровой сертификат и установить его на вашем сайте. Вы можете обратиться в один из сертификационных центров за цифровой подписью. В число наиболее известных коммерческих сертификационных центров входят такие компании, как VeriSign, CyberTrust и GTE. Вам необходимо заплатить за цифровую подпись. Сумма оплаты зависит от сертификационного центра и типа (класса) приобретаемого сертификата.

Так же, как и для торгового счета, на приобретение цифровой подписи требуется время. Вам следует обратиться за получением сертификата, когда вы начнете разрабатывать сайт электронной коммерции.

Механизм обеспечения безопасности передачи секретных данных

Для гарантии того, что сервер и соединение являются защищенными от неавторизованного доступа, вы можете использовать:

---

[1] Идентификация — присвоение пользователям, процессам, информационным ресурсам личного идентификатора и сравнение его с заданным перечнем. — *Пер.*

- протокол безопасных соединений (SSL, Secure Sockets Layer);
- протокол безопасных электронных транзакций (SET, Secure Electronic Transaction).

Рассмотрим, как каждая из этих опций обеспечивает безопасность вашего сервера и передачи данных.

Протокол Secure Sockets Layer представляет собой схему шифрования предназначенных для кодирования транзакций в таких протоколах, как HTTP, FTP и NNTP. Вы используете SSL вместе с цифровым сертификатом на вашем Web-сервере. Когда вы применяете SSL для передачи данных:

- происходит шифрование данных;
- между источником и сервером-получателем устанавливается безопасная связь;
- обеспечивается аутентификация сервера.

SSL также обеспечивает необязательную функциональную возможность по проверке идентичности (identity) клиента.

Для обработки данных кредитных карт вам необходимо разместить форму заказа на сервере, на котором установлена поддержка SSL. Web-страницы, требующие SSL для передачи данных, имеют URL, в начале которого вместо "http" стоит "https". Более подробную информацию о SSL вы можете получить по адресу: **http://msdn.microsoft.com/workshop/server/iis/websec.asp**.

Протокол Secure Electronic Transaction является открытым стандартом, разработанным совместно компаниями Microsoft, Netscape, MasterCard и Visa для обработки данных кредитных карт в Internet. SET использует комбинацию, состоящую из группы сертификационных центров, которые осуществляют аутентификацию каждого элемента транзакции: покупателя, компанию, выпускающую кредитные карты, вас, а также организацию (банк), предоставившую вам торговый счет. SET осуществляет разделение аналогично тому, как каждый из элементов транзакции "знает" лишь соответствующую ему часть в транзакции. Например, продавец будет знать лишь позиции заказа и суммарную заявленную стоимость, а также был ли принят платеж. Продавец не будет, однако, располагать какой-либо информацией о форме платежа, которую использовал клиент. Подобным же образом производитель кредитной карты покупателя будет знать суммарную заявленную стоимость и информацию о кредитной карте. Он не узнает о том, какие именно товары были куплены.

Далее приведены несколько советов по обеспечению безопасности сайта.

- Если необходимо хранить информацию о кредитных картах ваших клиентов, вам нужно хранить ее в базе данных в зашифрованном виде.
- Вам следует использовать брандмауэры (firewalls). Брандмауэр — это механизм обеспечения безопасности корпоративной сети от внешних ком-

муникационных сетей, таких как Internet. Он реализован с помощью компьютера, в котором установлено две сетевые интерфейсные карты и специальная программа брандмауэра. Одна карта соединена с корпоративной локальной вычислительной сетью, а другая подключена к Internet. Вся информация, передаваемая между этими двумя сетями, должна проходить сквозь этот барьер.

- Вам следует быть в курсе последних инструментальных средств и технологий в сфере безопасности. Для этого обращайтесь к бюллетеням по вопросам безопасности, опубликованным на сервере Microsoft Security Notification server (**http://www.microsoft.com/technet/security/notify.asp**) или на Web-сайте Microsoft Security Advisor Website (**http://www.microsoft.com /security/default.asp**).

Реализация безопасного интерфейса

Вы можете воплотить на практике опыт заказа "по щелчку мыши", воспользовавшись для этого одной из доступных на сегодняшний день служб электронных бумажников (wallet services). Эти службы позволяют покупателям хранить информацию о себе в одном безопасном месте. Когда покупатель посещает сайт электронной коммерции и покупает товары, ссылка на форму (бланк) покупки или форму заказа позволяет извлекать информацию о покупателе непосредственно на центральном сервере. Примером такой службы является Microsoft Passport.

Этапы обработки данных кредитной карты

Программное обеспечение обработки платежей должно быть очевидным (понятным) для покупателя и владельца сайта. Вы должны использовать этот пакет программ, добавив соответствующий код в форму заказа. Перед установкой этого программного обеспечения на сайте вам необходимо определить, хотите ли вы обрабатывать и отображать данные реального времени, или же обрабатывать данные кредитных карт в онлайновом режиме. Вам может понадобиться обрабатывать данные реального времени для продажи информации и программного обеспечения в онлайновом режиме, или для отображения аутентифицированных страниц.

Обработка данных кредитной карты состоит из трех этапов:

1. Авторизация (authorization).
2. Прием транзакции (capture).
3. Возмещение (refund).

Рассмотрим каждый из этих этапов.

Этап авторизации

Этап авторизации начинается с того момента, когда программное обеспечение обработки платежей принимает информацию о кредитной карте для обработки. На этом этапе программное обеспечение обработки платежей выполняет следующие задачи:

1. Осуществляет проверку информации о кредитной карте.
2. Проверяет, имеется ли на счету достаточно средств для оплаты заказанных товаров.

К тому же на кредитной карте выделяется резерв (hold) для заявленной стоимости. Этот резерв удерживается до тех пор, пока товары не будут реально доставлены покупателю.

Между этапом авторизации и этапом сбора данных продавец должен выполнить заказ и доставить товар. Если товар не был отгружен до истечения этапа авторизации, вам, возможно, потребуется выполнять возврат. Это может повлечь дополнительные расходы на выполнение транзакций.

Прием транзакции

Этап приема транзакции начинается, когда продавец отгружает товары и отправляет запрос на прием транзакции. На этом этапе банк оценивает информацию авторизации и передает фонды из банка покупателя на ваш счет.

Для онлайновых служб, которые представляют собой аутентифицированные страницы с непосредственным доступом, или же загружаемое по сети ("скачиваемое") программное обеспечение, этапы авторизации и приема транзакции совпадают по времени.

Если заказ включает в себя несколько наименований (позиций), и лишь некоторая часть из них доставлена, вы можете отправить запрос на прием транзакции только для тех товаров, которые были отгружены. В результате часть заявленной суммы будет переведена на торговый счет. Вам, возможно, придется провести повторную авторизацию для оставшейся суммы, относящейся к товарам, которые не были отгружены. Это предполагает дополнительные издержки.

Этап возмещения

Покупатель может отменить заказ или же вернуть доставленные ему товары. В таких случаях вам необходимо возместить оплату заказа. Этап возмещения является противоположным по отношению к этапу приема транзакции. На этом этапе вы предоставляете авторизационный номер (authorization number), а также сумму, подлежащую возврату. В ответ на запрос о возмещении, процессор возвращает фонды, которые были переведены на торговый счет.

Реализация обработки данных кредитной карты

В следующих разделах будет рассмотрена:

- реализация обработки данных кредитной карты на сайте, оснащенном Site Server Commerce Edition 3.0;
- реализация Microsoft Passport на сайте.

Интеграция обработки данных кредитных карт с Site Server Commerce Edition 3.0

Для осуществления интеграции обработки данных кредитных карт с Site Server Commerce Edition 3.0 вам необходимо выполнить следующие задачи:

1. Использовать компонент конвейера для обработки заказов по кредитной карте.
2. Создать файл конфигурации конвейера, который осуществит привязку данного компонента к ступени конвейера.

Если вы пользуетесь услугами фирмы по обработке платежей, эта фирма предоставит вам программный компонент, который необходимо встроить в Microsoft Commerce Pipeline. Например, пакет CyberSource имеет в своем составе объект `Commerce`, который вы можете легко встроить в Microsoft Commerce Pipeline как часть архитектуры электронной коммерции Site Server.

Вы можете вставить этот компонент в конвейер с помощью Microsoft Commerce Pipeline Editor. Указанный редактор также предоставит вам возможность создать конфигурационный файл, соответствующий различным ступеням в данном компоненте. Объект конвейера загружает этот файл перед выполнением конвейера.

Обычно вам следует добавлять программный компонент обработки платежей в ступень оплаты конвейера закупок (Purchase pipeline). К тому же необходимо написать код для авторизации, приема транзакций и возмещения, в котором в качестве ввода будет использоваться объект `mscsOrderForm`.

Давайте теперь рассмотрим код, который используется для авторизации. В этом коде мы добавили информацию о покупателе и кредитной карте в форму заказа и запустили конвейер авторизации (auth.pcf) с помощью данного компонента:

```
<%
Function Authorization(mscsOrderForm)
  Dim mscsPipeContext, errorLevel
  ' Установка контекста конвейера
  mscsPipeContext("Language") = Request.Form("country")
```

```
  mscsOrderForm.Value("cc_name") = Request.Form("creditc_name")
  mscsOrderForm.Value("cc_type") = Request.Form("creditc_type")
  mscsOrderForm.Value("_cc_number") = Request.Form("creditc_number")
  mscsOrderForm.Value("_cc_expmonth") = Request.Form("creditc_expmonth")
  mscsOrderForm.Value("_cc_expyear") = Request.Form("creditc_expyear")
  mscsOrderForm.Value("bill_name") = Request.Form("bill_name")
  mscsOrderForm.Value("bill_phone") = Request.Form("bill_phone")
  mscsOrderForm.Value("bill_street") = Request.Form("bill_street")
  mscsOrderForm.Value("bill_city") = Request.Form("bill_city")
  mscsOrderForm.Value("bill_state") = Request.Form("bill_state")
  mscsOrderForm.Value("bill_zip") = Request.Form("bill_zip")
  mscsOrderForm.Value("bill_country") = Request.Form("bill_country")
  mscsOrderForm.Value("_shopper_first_name") = Request.Form("first_name")
  mscsOrderForm.Value("_shopper_last_name") = Request.Form("last_name")
  mscsOrderForm.Value("_shopper_email") = Request.Form("email")
  mscsOrderForm.Value("_shopper_phone") = Request.Form("phone")
  mscsOrderForm.Value("_shopper_street") = Request.Form("street1")
  mscsOrderForm.Value("_shopper_city") = Request.Form("city")
  mscsOrderForm.Value("_shopper_state") = Request.Form("state")
  mscsOrderForm.Value("_shopper_zip") = Request.Form("zip")
  mscsOrderForm.Value("_shopper_country") = Request.Form("country")
  mscsOrderForm.Value("__total_total") = Request.Form("total_Amount")
  ' Создание объекта конвейера и загрузка конфигурационного файла
  set pipeline = Server.CreateObject("Commerce.MtsPipeline")
  ' Установка файлов конфигурации и регистрации
  Call pipeline.LoadPipe(d:\inetpub\wwwroot\fivelakes\config\auth.pcf)
  Call pipeline.SetLogFile("c:\temp\flpipeline.log")
  ' Запуск конвейера для авторизации
  errorLevel = pipeline.Execute(1, mscsOrderForm, pipeContext, 0)
  ' В зависимости от результата вы можете запустить этот процесс снова
  Authorization = errorLevel
end function
%>
```

В следующей функции используются объекты `orderform` и `order_number` для ввода и выполнения приема транзакции. Объект `order_number`[1] является номером авторизации, который требуется для данного этапа.

```
<%
Function Capture(mscsOrderForm)
  Dim mscsPipeContext, errorLevel
  mscsOrderForm.Value("_total_total") = Request.Form("amount_capture")
  mscsOrderForm.Value("_shopper_bill_orderno") = "Your order number"
```

[1] В коде этот объект имеет имя `_shopper_bill_orderno`. — *Ред.*

```
  mscsOrderForm.Value("_shopper_first_name") = Request.Form("first_name")
  mscsOrderForm.Value("_shopper_last_name") = Request.Form("last_name")
  mscsOrderForm.Value("_shopper_email") = Request.Form("email")
  mscsOrderForm.Value("_shopper_phone") = Request.Form("phone")
  mscsOrderForm.Value("_shopper_street") = Request.Form("street1")
  mscsOrderForm.Value("_shopper_city") = Request.Form("city")
  mscsOrderForm.Value("_shopper_state") = Request.Form("state")
  mscsOrderForm.Value("_shopper_zip") = Request.Form("zip")
  mscsOrderForm.Value("_shopper_country") = Request.Form("country")
  ' Установка контекста конвейера
  mscsPipeContext("Language") = Request.Form("usa")
  ' Создание объекта конвейера и загрузка конфигурационного файла
  set pipeline = Server.CreateObject("Commerce.MtsPipeline")
  ' Установка файлов конфигурации и регистрации конвейера
  Call pipeline.LoadPipe(d:\inetpub\wwwroot\fivelakes\config\capture.pcf)
  Call pipeline.SetLogFile("c:\temp\flpipeline.log")
  ' Запуск конвейера для приема транзакции
  errorLevel = pipeline.Execute(1, mscsOrderForm, pipeContext, 0)
  ' В зависимости от результата вы можете запустить этот процесс снова
  Capture = errorLevel
End Function
%>
```

В процессе возмещения также используется объект `orderform` в качестве ввода. Вы добавляете информацию о покупателе, а также информацию об авторизации и запускаете данный компонент, как представлено в следующем коде:

```
<%
Function Refund(mscsOrderForm)
  Dim mscsPipeContext, errorLevel
  mscsOrderForm.Value("_total_total") = Request.Form("amount_refund")
  mscsOrderForm.Value("_shopper_bill_orderno") = "Your order number"
  mscsOrderForm.Value("_shopper_first_name") = Request.Form("first_name")
  mscsOrderForm.Value("_shopper_last_name") = Request.Form("last_name")
  mscsOrderForm.Value("_shopper_email") = Request.Form("email")
  mscsOrderForm.Value("_shopper_phone") = Request.Form("phone")
  mscsOrderForm.Value("_shopper_street") = Request.Form("street1")
  mscsOrderForm.Value("_shopper_city") = Request.Form("city")
  mscsOrderForm.Value("_shopper_state") = Request.Form("state")
  mscsOrderForm.Value("_shopper_zip") = Request.Form("zip")
  mscsOrderForm.Value("_shopper_country") = Request.Form("country")
  ' Установка контекста конвейера
  mscsPipeContext("Language") = Request.Form("usa")
  ' Создание объекта конвейера и загрузка конфигурационного файла
  set pipeline = Server.CreateObject("Commerce.MtsPipeline")
```

```
  ' Установка файлов конфигурации и регистрации конвейера
  Call pipeline.LoadPipe(d:\inetpub\wwwroot\fivelakes\config\refund.pcf)
  Call pipeline.SetLogFile("c:\temp\flpipeline.log")
  ' Запуск конвейера для возврата средств
  errorLevel = pipeline.Execute(1, mscsOrderForm, pipeContext, 0)
  ' В зависимости от результата можно запустить эту процедуру снова
  Refund = errorLevel
End Function
%>
```

Применение Microsoft Passport

Изучение поведения при осуществлении покупок в Internet показало, что покупатели прекращают транзакции в ответ на просьбу заполнить форму с персональными данными и ввести информацию о кредитной карточке. Вы можете устранить эту проблему с помощью Microsoft Passport на своем Web-сайте.

Что такое Microsoft Passport?

Microsoft Passport представляет собой набор из двух служб, поддержку которых осуществляет Microsoft:

- служба Single Sign-In (SSI) предоставляет возможность ввода одного имени и пароля для входа на большое количество Web-сайтов.
- служба электронного бумажника (Wallet Service) помогает клиенту осуществлять быструю покупку товаров и служб посредством Internet.

Рассмотрим каждую из этих служб.

Служба SSI

Microsoft Passport позволяет покупателям посещать все сайты, на которых установлена эта служба, по одному имени входа и паролю. Необходимость указания персональной информации при этом возникает лишь один раз. Члену Microsoft Passport назначается уникальный паспортный идентификатор (Passport ID), который ассоциируется с пользователем службы, а не с компьютером. Следовательно, пользователи этой службы могут посещать сайты, на которых установлена эта служба, с любого компьютера.

SSI также дает вам возможность обслуживать каждого заказчика индивидуально через Web на основе паспортного идентификатора.

Служба Wallet

Passport Wallet хранит номера кредитных карт своих членов, а также адреса оплаты и отгрузки товара. Эта информация поступает в совместное исполь-

зование среди сайтов — участников службы, только тогда, когда сайт, являющийся членом службы Passport, запрашивает ее на сайте, авторизованном корпорацией Microsoft в качестве так называемого торговца службы Passport (Passport merchant).

Служба Passport Wallet обладает высокой степенью безопасности, т. к. использует протоколы HTTP и SSL для отправки содержащихся в форме данных методом POST на сайт. К тому же покупателям нет необходимости инсталлировать какое-либо клиентское программное обеспечение, поскольку Microsoft Passport размещается на сервере.

Преимущества Microsoft Passport

Если вы будете использовать Microsoft Passport на своем Web-сайте, то вы сможете:

- повысить объем продаж и регистраций;
- увеличить покупательские сборы (customer acquisition) и удержания (retention).

Покупателям не придется многократно вводить свои персональные данные. Покупатели ценят простоту и безопасность посещения и покупок на сайтах, использующих Microsoft Passport.

Для пользователя сайта Microsoft Passport является:

- Простым в использовании.

  Microsoft Passport присваивает каждому покупателю одно имя и пароль для входа на Web-сайт и доступа к его службам. Затем, пользователи могут войти на другие Web-сайты, участвующие в системе Microsoft Passport, по одному щелчку мыши.

- Быстрым.

  Microsoft Passport предоставляет покупателям электронные бумажники, в которых они могут указывать информацию о себе. Эти сведения включают персональную информацию и данные о кредитной карте. Покупатели могут быстро делать покупки в сети, используя свою индивидуальную информацию (profile information) и бумажники, не прибегая к ручному вводу информации.

- Безопасным.

  Индивидуальная информация и данные бумажника хранятся в зашифрованном виде и защищены строгой политикой безопасности. К тому же личная информация удаляется из компьютера сразу по выходу. Таким образом, покупатели чувствуют себя в безопасности, используя Microsoft Passport на компьютерах с разделенным и общим доступом.

Реализация Microsoft Passport на сайте

Для того чтобы реализовать обе службы, входящие в Microsoft Passport на вашем сайте, вам необходимо прежде всего зарегистрировать свой сайт.

Ниже приведены требования к аппаратному и программному обеспечению, необходимые для реализации службы Single Sign-In на вашем Web-сайте.

Требования к аппаратному обеспечению:

- x86 компьютер с процессором Pentium;
- 64 Мбайт RAM;
- устройство CD-ROM;
- сетевая карта с уникальным фиксированным IP-адресом.

Требования к программному обеспечению:

- Windows 2000 и IIS 5.0, Windows NT 4.0 с Service Pack 3.0 и выше, IIS 4.0, и IE 4.0 и выше;
- сертификаты SSL.

Служба Single Sign-In

Для того чтобы реализовать службу Single Sign-In на Web-сайте, вам необходимо инсталлировать Passport Manager, входящий в состав Passport Single Sign-In SDK. Затем вы можете совершенствовать и тестировать свой Web-сайт, добавив в него серверный скрипт и элементы пользовательского интерфейса.

Вам необходимо написать код для сбора информации и удаления cookies по выходу с сайта.

Следующий код будет удалять cookies, относящиеся к Passport при выходе с сайта. Вам нужно написать этот код в файле с именем logoutgif.asp.

```
<%
  ' Освобождаем кэш браузера:
  Response.Expires=-1
  Response.AddHeader "Cache-Control", "no-cache"
  Response.AddHeader "Pragma", "no-cache"
  ' Удаление двух cookies, относящихся к Microsoft Passport server
  Response.Cookies("MicrosoftPProf") = ""
  Response.Cookies("MicrosoftPAuth") = ""
  ' Действия cookie истекает немедленно
  Response.Cookies("MicrosoftPProf").Expires = #Jan 1,2002#
  Response.Cookies("MicrosoftPAuth").Expires = #Jan 1,2002#
  ' Удаление всех cookies, относящихся к сайту Ramona
  Response.Cookies("Ramona") = ""
```

```
  Response.Cookies("Ramona").Expires = #Jan 1,2002#
%>
<!— На завершающей работу на сайте странице после имени сайта ->
<!- будет демонстрироваться этот рисунок ->
<!—#include file="signoutcheckmark.gif"—>
```

Файл signoutcheckmark.gif с графическим изображением должен быть помещен в ту же папку, в которой находится файл logoutgif.asp, для того чтобы по выходу с сайта на странице Microsoft Passport server демонстрировалось соответствующее изображение, служащее в качестве подтверждения выхода с сайта.

Служба Passport Wallet

Для реализации службы Passport Wallet вам необходимо поместить специальные кнопки Passport Express Purchase, Passport Wallet или текстовую ссылку на страницах заказа вашего Web-сайта. Эти ссылки дадут вам возможность извлечь информацию из пользовательского электронного бумажника и отправить ее в виде формы по методу POST через соединение, защищенное протоколом SSL. Затем вам необходимо добавить ASP-скрипт для извлечения отправленной по методу POST информации, принимаемой на сайте.

В дополнение к этим обязательным шагам вы можете сделать следующее:

- поместить на страницах Microsoft Passport свой текст и графику;
- настроить страницы;
- использовать CSS для определенных шрифтов и цвета фона.

Лабораторная работа 11. Обработка данных кредитных карт

Цели и задачи

После выполнения этой лабораторной работы вы должны уметь реализовать возможности Microsoft Passport на сайте.

Перед началом работы

Перед началом работы вы должны иметь:

- базовые знания HTML и ASP-скриптов;
- хорошие знания Microsoft Internet Explorer и Microsoft Visual InterDev 6.0;
- доступ к Internet.

Подготовка к работе

Для выполнения этой работы необходимо скопировать файлы из папки Labs\Lab11\Ramona\StartCode в папку InetPub\wwwroot\Ramona.

Примечание

Время, необходимое для выполнения данной работы, — 30 минут.

Упражнение 1. Реализация Microsoft Passport на сайте

В этом упражнении вы установите службу Microsoft Passport на сайте Ramona.

Для установки Microsoft Passport на Web-сервере необходимо:

1. Переписать файл passport_sdk_v1_1.exe с сайта **http://www.passport.com /business/sdk.asp** в папку C:\TEMP. Для этого нужно войти в качестве члена Microsoft Passport.
2. С помощью Windows NT Explorer перейти в каталог C:\TEMP.
3. Дважды щелкнуть по значку файла passport_sdk_v1_1.exe для того, чтобы он разархивировался.
4. В качестве папки для разархивированных файлов указать C:\TEMP.
5. Дважды щелкнуть по значку файла Setup.exe в папке C:\TEMP.
6. Запустится программа установки. Щелкните кнопку **Next**, а затем кнопку **Yes**.
7. Введите ключ программного продукта, а потом нажмите кнопку **Next**. Ключ отображался на странице, с которой был переписан файл passport_sdk_v1_1.exe.
8. В диалоговом окне **Choose Destination Location** щелкните по кнопке **Next** для того, чтобы подтвердить место для инсталляции Microsoft Passport, заданное по умолчанию.
9. В диалоговом окне **Setup Type** выберите переключатель **Custom** и нажмите кнопку **Next**.
10. В диалоговом окне **Select Components** подтвердите значения, принятые по умолчанию, и щелкните по кнопке **Next**.
11. В диалоговом окне **Select Program Folder** подтвердите выбор по умолчанию и нажмите кнопку **Next**.
12. Выберите **Default Web Site**, а затем щелкните по кнопке **Next**. Программа установки прервет работу службы World Wide Web Publishing Service, инсталлирует файлы, зарегистрирует динамические библиотеки и вновь запустит World Wide Web Publishing Service.

13. Программа установки предложит перезагрузить компьютер. Щелкните кнопку **Finish** для перезагрузки.

Для реализации Microsoft Passport на сайте Ramona:

1. Запустите Windows NT Explorer и перейдите в папку Inetpub\wwwroot \Ramona.
2. Для включения функциональных возможностей MS Passport на сайте откройте файл payment.asp. Для этого щелкните правой кнопкой мыши по значку файла payment.asp и затем выберите команду **Edit** или **Open**.
3. Запустите работу cookies браузера для файла payment.asp, введя следующий код вверху страницы:

```
<%
  ' Отменить кэширование страницы в браузере
  Response.Expires = 0
  Response.AddHeader "Cache-Control", "no-cache"
  Response.AddHeader "Pragma", "no-cache"
%>
```

4. Создайте экземпляр объекта `Passport Manager`, введя представленный ниже код после того, что был набран в п. 3:

```
<%
  Dim oPassMgrObj
  Set oPassMgrObj = Server.CreateObject ("Passport.Manager.1")
%>
```

5. Установите параметры объекта `Passport Manager`, добавив следующий код после того, что был введен в п. 4:

```
<%
  Dim ThisPageURL, TimeWindow, ForceLogin
  Dim CoBrandArgs, LangID, Secure
  ThisPageURL = "http://" & Request.ServerVariables ("SERVER_NAME")
  ThisPageURL = ThisPageURL & Request.ServerVariables ("SCRIPT_NAME")
  TimeWindow=600     ' В секундах, 10 минут
  ForceLogin=False   ' Для обновления cookie пользователю
                     ' не надо вводить информацию
  CoBrandArgs=False  ' Передачи совместных аргументов нет
  LangID=1033        ' По умолчанию английский
  ' Проверка использования переменной сервера объекта Request по SSL
  If Request.ServerVariables ("HTTPS") = "on" Then
    Secure = TRUE
  Else
    Secure = FALSE
  End If
%>
```

6. Инициируйте отображение пользовательского интерфейса для входа в службу Passport (SignIn) и выхода из нее (SignOut), введя следующий код в начале страницы:

```
<% = oPassMgrObj.LogoTag (Server.URLEncode (ThisPageURL), _
     TimeWindow, ForceLogin, CoBrandArgs, LangId, Secure) %>
```

7. Сохраните страницу payment.asp.

Для перезагрузки сайта в административной консоли Site Server (Site Server MMC):

1. На панели задач нажмите кнопку **Start**, укажите пункт **Programs**, а затем команду **Microsoft Site Server**.
2. Укажите **Administration**, а затем щелкните по значку **Site Server Service Admin (MMC)**.
3. Щелкните по знаку +, следующему за значком **Commerce Host Administration**.
4. Щелкните правой кнопкой мыши ***<localhost>*** под значком **Commerce Host Administration**, выберите команду **All Tasks**, а затем команду **Reload All Commerce Sites**.

Вопросы для повторения

1. Что требуется для реализации SSL на сервере?
2. В чем отличие этапа авторизации от этапа приема транзакции при обработке данных кредитной карты?
3. Как осуществляется интегрирование программного обеспечения обработки платежей на сайте Site Server Commerce Edition?
4. Как Microsoft Passport гарантирует безопасность конфиденциальных данных?

ГЛАВА 12

Настройка сайтов в соответствии с потребностями пользователей

Краткий обзор

В составе Microsoft Site Server Commerce Edition 3.0 имеются специальные программы — мастера для создания Web-сайтов электронной коммерции. Вероятно, вам захочется модифицировать функциональные возможности сайта в соответствии с потребностями своего бизнеса. В этой главе вы узнаете, как выполняется настройка сайтов, созданных с помощью программы Site Builder. Вы также научитесь модифицировать ASP-страницы, из которых состоит ваш сайт.

После изучения данной главы вы сможете:

- осуществлять модификацию сайта Microsoft Site Server Commerce Edition 3.0;
- модифицировать ASP-страницы для реализации ими дополнительных функциональных возможностей.

Модификация сайтов

Рассмотрим следующие вопросы:

- смена папки сайта;
- модификация атрибутов и вариантов продукта.

Сначала выясним, как осуществляется смена папки сайта.

Смена папки сайта

Как правило, сайт следует создавать на тестовом сервере. Перед эксплуатацией вам потребуется переместить сайт в иное место. Это может быть дру-

гой сервер или другая папка на том же самом сервере. К тому же сайт может быть запущен в собственном домене. Так или иначе, вам необходимо перемещать сайт в другую папку. Для этого нужно выполнить следующие действия:

1. Создать новую папку на сайте
2. В этой папке сформировать новый сайт.
3. Создать "фундамент" (foundation) нового сайта.
4. Скопировать файлы существующего сайта в новый каталог, замещая ими те, что уже имеются в новом каталоге.
5. Закрыть предыдущий ("старый") сайт и запустить новый.
6. Модифицировать используемые на сайте файлы конфигурации конвейера.

Сайт электронной коммерции содержит компоненты конвейера. В свою очередь, компонент конвейера может включать в себя компонент `Scriptor`, который использует внешний скрипт. Если вы производите смену папки сайта, необходимо отредактировать файл скрипта, указав в нем, с учетом нового местоположения, правильный путь.

Ниже представлен пример файла скрипта для сайта FiveLakes. Этот файл был перемещен в папку d:\newfivelakes\newfivelakes. Для нового сайта изменилось местоположение файла конфигурации CIT-конвейера (Commerce Interchange Transmit pipeline).

```
<%
Function mscsexecute(config, orderform, context, flags)
  mscsexecute = 1

  Dim mscsMtsTxPipeline

  Set mscsMtsTxPipeline = CreateObject("Commerce.MtsTxPipeline")
  call mscsMtsTxPipeline.loadpipe _
    ("d:\newfivelakes\newfivelakes\config\transmit.pcf")

  Dim TransportDictionary

  Set TransportDictionary = CreateObject("Commerce.Dictionary")
  Set TransportDictionary.object = orderform
  TransportDictionary.Receipt_requested = "yes"
  errorlevel = mscsMtsTxPipeline.Execute _
    (1, TransportDictionary, context, 0)
End Function
%>
```

Модификация вариантов и атрибутов продукта

На сайте электронной коммерции, в схеме базы данных имеются так называемые *продукты* (products). В качестве продукта может быть конкретный *элемент* (item) или множество непосредственно связанных элементов. У продукта есть атрибуты. Атрибут может содержать единственное значение или множество значений.

Например, на Web-сайте Ramona осуществляется продажа книг. Следовательно, семейство продуктов на сайте представляют книги. В качестве атрибутов могут выступать автор или тематика издания, например, "Компьютеры и развлечения". Вариант продукта (product variant) — это сочетание его атрибутов. Каждому продукту назначен отдельный признак — *единица учета запасов* (Stock-Keeping Unit, SKU).

Можно изменять атрибуты продукта и помещать варианты продуктов на сайте с помощью Site Builder Wizard (Мастера компоновки сайта). Варианты продуктов можно также размещать на заказных сайтах (custom-built sites), используя для этого Microsoft Site Server Commerce Edition 3.0.

Существует возможность включать дополнительные атрибуты продукта в рекламно-коммерческих целях. Например, вы хотите предложить объемную скидку (bulk discount) на книги по компьютерной тематике, написанные каким-то конкретным автором. Это можно сделать, включив еще одну пару параметров "ключ/значение" в объект словаря данного продукта в ступени Product Info, принадлежащей P-конвейеру. Вам потребуется компонент `Scriptor`, который запишет новое значение в форму заказа в виде параметра `_product_attribute`. Это значение затем будет использоваться для учета рекламно-коммерческих скидок (promotional discounts).

Настройка страниц

Далее мы рассмотрим следующие вопросы:

- вызов отдельного компонента конвейера на странице;
- ограничение доступа к страницам (предоставление доступа зарегистрированным пользователям);
- подтверждение покупок по электронной почте.

В дополнение к настройке сайта вы можете модифицировать его отдельные ASP-страницы.

Это обновление ASP-страниц может заключаться в инициировании на странице вызова компонентов конвейера, организации управления доступом к страницам, а также в отправке сообщений электронной почты клиентам

вашего сайта. Сначала рассмотрим вопрос о вызове компонента на ASP-странице.

Вызов отдельного компонента конвейера на ASP-странице

Как вы уже знаете, конвейер состоит из различных компонентов, которые выполняются каждый раз, когда вы загружаете и запускаете конвейер. В некоторых случаях вам может потребоваться запустить не весь конвейер, а лишь отдельные его компоненты. Напрямую из ASP-страницы это сделать нельзя, т. к. в компонентах реализуются методы и интерфейсы, которые не поддерживаются в скрипт-языке Microsoft Visual Basic Scripting Edition. Однако вы можете выполнить компонент, создав только для него специальный конвейер, который потом можно запускать. Подобный подход практически осуществим, но он влияет на производительность сайта. Например, каждый раз при запуске конвейера необходимо загружать его полную конфигурацию, что требует дополнительных ресурсов.

Для решения этой проблемы в Site Server Commerce Edition 3.0 предусмотрен специальный компонент `MicroPipe`, позволяющий осуществлять запуск и выполнение отдельного компонента из ASP-страницы, без привлечения всего конвейера, с помощью pcf-файла.

Компонент `MicroPipe` обеспечивает следующие методы:

- метод `SetComponent` используется для указания компонента, подлежащего загрузке и выполнению;
- метод `Execute` запускает компонент конвейера, указанный в методе `SetComponent`;
- метод `SetLogFile` указывает файл для регистрации выполнения операций.

Теперь рассмотрим, как компонент `MicroPipe` используется на ASP-странице. Вам необходимо сначала создать и установить компонент, который будет выполняться с помощью `MicroPipe`. Для этого:

1. Создайте экземпляр компонента `MicroPipe` с именем `mpcomp`:

```
Set mpcomp = Server.CreateObject("Commerce.MicroPipe")
```

2. Создайте экземпляр компонента конвейера, который вам необходимо запустить на ASP-странице. Например, с помощью следующего кода вы создадите экземпляр компонента `MyPipeline`:

```
Set objPipe = Server.CreateObject("MyPipeline")
```

3. Передайте объект, сформированный на предыдущем шаге, в метод `SetComponent` компонента `MicroPipe`. В данном примере вы можете ввести следующий код:

```
mpcomp.SetComponent(objPipe)
```

4. Запустите компонент с помощью метода `Execute` компонента `MicroPipe`. Для этого используется следующий синтаксис:

```
MicroPipe.Execute(MainDictionary, Context, Reserved)
```

Здесь:

- `MicroPipe` указывает имя объекта `MicroPipe`;
- `MainDictionary` задает главный словарь конвейера, такой как, например, объект `orderform`;
- `Context` указывает на содержимое конвейера;
- `Reserved` не используется. Значение этого параметра должно быть 0.

5. Проверьте результат операции. Метод `Execute` возвращает следующие значения типа `Long`:
   - 1 — успешное выполнение конвейера;
   - 2 — ошибки, вызванные компонентами конвейера, содержащимися в коллекциях ошибок Basket ("Корзина") или ошибок при покупке (purchase error). Это ошибки покупателя, например, пропуск информации;
   - 3 — фатальная ошибка конвейера. Она может быть вызвана тем, что конвейер не получает доступ к базе данных.
6. Произведите очистку объектов с помощью следующего кода:

```
Set mpcomp = nothing
Set objpipe = nothing
```

Ограничение доступа к страницам

Часть содержимого (content) вашего сайта может предназначаться только для зарегистрированных пользователей. Вы можете отредактировать ASP-страницы, включив в них ограничения, чтобы к ним смогли обращаться только зарегистрированные пользователи.

Использование идентификатора покупателя для ограничения доступа

Для ограничения доступа к странице вы должны извлечь значение так называемого уникального идентификатора покупателя — `ShopperID`. Если оно равно `NULL`, то это означает, что посетитель сайта не зарегистрирован. Затем вы можете "направить" этого пользователя на страницу default.asp или же на страницу регистрации. Вот пример кода, который вы можете включить в страницы, предназначенные только зарегистрированным пользователям (в идеале, вам следует поместить представленный код в include-файл и вызывать этот файл в каждой странице ASP-приложения):

```
<%
mscsShopperID = mscsPage.GetShopperId
if IsNull(mscsShopperID) then
  if Not this_page("default.asp") then
    Response.Redirect(mscsPage.URL("default.asp"))
  end if
end if

%>
```

В этом коде для извлечения значения идентификатора `ShopperID` используется метод `GetShopperID` объекта `Page`.

Хранение "конфиденциальных" страниц в отдельной папке

Еще один метод ограничения доступа к определенным страницам заключается в хранении их в отдельном каталоге. Для придания большей интерактивности сайту вы опять-таки можете направить посетителя на страницу регистрации. Затем вы можете воспользоваться написанным вами кодом, который определит местоположение запрашиваемого пользователем файла. Этот же код сможет проверить, является ли данный пользователь зарегистрированным пользователем. Зарегистрированные пользователи получают доступ к "конфиденциальным" (restricted) страницам.

Ниже представлен пример кода, осуществляющего проверку того, что вызванная страница находится в каталоге RegMembers. Если новый пользователь попытается получить доступ к какой-либо странице в этом каталоге, он будет направлен на страницу регистрации — register.asp.

```
<%
if IsNull(mscsShopperId) then
  targetPage = Request.ServerVariables("SCRIPT_NAME")
  virtualFolderMember = "/" & mscsPage.VirtualFolder & "/regmember"
  if Left(targetPage, len(virtualFolderMember)) = virtualFolderMember
then
    redirectPage = targetPage & "?"
    query = Request.ServerVariables("QUERY_STRING")
    body = Request.Body
    if Not(IsNull(query)) then
      redirectPage = redirectPage & query
    end if
    if Not(IsNull(body)) then
      redirectPage = redirectPage & body
    end if
    Response.Redirect(mscsPage.URL("register.asp", "redirect_target",
redirectPage))
```

```
    Response.End
  end if
else
end if
%>
```

Подтверждение покупок по электронной почте

Было бы хорошо, чтобы покупатель, делая заказ, всякий раз получал подтверждение о нем по электронной почте. Такое подтверждающее сообщение может содержать информацию о заказе и контрольно-регистрационный номер заказа (order-tracking number).

Для того чтобы включить в ASP-приложение такую функциональную возможность, как отправка сообщений электронной почты, вы можете использовать библиотеку объектов данных для поддержки совместной работы — Microsoft CDO for NTS Library (Collaboration Data Objects for Windows NT Server Library), версия 1.2. В этой библиотеке представлены объекты, которые вы можете использовать для обеспечения отправки сообщений из приложений Microsoft Visual Basic, Microsoft Visual C++, C/C++, и Visual Basic Scripting Edition (VBScript).

Чтобы создать сообщение электронной почты и отправить его покупателю, вам нужно использовать объект `NewMail` библиотеки CDO for NTS Library. Этот объект позволяет отправлять сообщения без необходимости регистрации во время сеанса.

Прежде чем приступить к рассмотрению использования на сайте объекта `NewMail`, давайте обсудим его методы и свойства. Основными свойствами объекта `NewMail` являются:

- `To` — полный адрес электронной почты получателя объекта `NewMail`;
- `Cc` — полный адрес электронной почты получателя копии объекта `NewMail`;
- `Bcc` — полный адрес электронной почты получателей скрытой копии (blind copy) объекта `NewMail`;
- `From` — адрес электронной почты отправителя объекта `NewMail`;
- `Subject` — тема посылаемого сообщения;
- `Body` — текст почтового сообщения (может быть представлен в формате простого текста или HTML);
- `Importance` — степень важности отправления объекта `NewMail`;
- `Value` — дополнительный заголовок (header) почтового сообщения.

Теперь рассмотрим методы объекта NewMail:

- AttachFile — позволяет вам присоединять файл к почтовому сообщению;
- AttachURL — позволяет присоединять данные к сообщению и связывать их с URL;
- Send — осуществляет отправку объекта NewMail получателям, указанным свойствами To, Cc и Bcc объекта NewMail.

Теперь подробно рассмотрим метод Send. Он имеет следующий синтаксис:

```
ObjNewMail.Send([From] [,To] [,Subject] [,Body] [,Importance])
```

Параметры метода Send соотносятся с соответствующими свойствами объекта NewMail. Однако, если вы укажете значение свойства To, а также значение параметра To, объект NewMail будет отправлен обоим получателям.

Как только метод Send будет успешно выполнен, объект NewMail станет недействительным и не сможет использоваться повторно.

Теперь ознакомимся с тем, как отправить подтверждающее сообщение покупателю, сделавшему заказ. Для этого вам нужно вставить объект Scriptor в конвейер закупок (Purchase pipeline)[1] в OPP и ассоциировать (associate) vbs-файл с компонентом Scriptor. Файл с расширением vbs будет содержать код, который создает экземпляр объекта NewMail и отправляет почту указанному получателю.

Здесь представлен код, содержащийся в файле email.vbs, создающий объект NewMail и отсылающий этот объект указанному получателю.

```
Function mscsexecute(config, orderform, context, flags)
  Dim myMail
  Set FiveLakesMail = CreateObject("CDONTS.NewMail")
  FiveLakesMail.From = "Service@FiveLakes.com"
  FiveLakesMail.To = orderForm.ship_to_email
  FiveLakesMail.Subject = "Your order has been confirmed. Thank you. _
     (Подтверждаем получение Вашего заказа. Спасибо.)
  FiveLakesMail.Body = "This is a sample message.(Это пример сообщения.)
  FiveLakesMail.Send
  Set FiveLakesMail = Nothing
End function
```

После создания файла email.vbs с представленным кодом вы открываете конвейер закупок (Purchase pipeline) сайта FiveLakes в редакторе Microsoft Commerce Pipeline Editor. В этот конвейер вы вставляете объект Scriptor и ассоциируете файл email.vbs с объектом Scriptor.

---

[1] Авторы не точны. Purchase pipeline не входит в OOP. Его функции в классе B2B выполняет CPP-конвейер *(см. главу 3)*. — *Пер.*

Лабораторная работа 12. Настройка сайта

Цели и задачи

После выполнения этой лабораторной работы вы должны уметь создавать сообщение электронной почты с помощью компонента `Scriptor`.

Перед началом работы

Перед началом работы вы должны иметь:

- базовые знания HTML и ASP-скриптов;
- хорошие знания Microsoft Internet Explorer и Microsoft Visual InterDev 6.0.

Подготовка к работе

Для выполнения этой работы необходимо скопировать файлы из папки Labs\Lab12\FiveLakes\StartCode в папку Inetpub\wwwroot\FiveLakes\Config

Примечание

Время, необходимое для выполнения данной работы, — 15 минут.

Упражнение 1. Создание сообщения электронной почты с помощью компонента *Scriptor*

В этом упражнении вы сконфигурируете сайт FiveLakes для отправки с помощью электронной почты подтверждения о получении заказа покупателю.

Чтобы создать сценарий для компонента `Scriptor`:

1. Откройте Windows NT Explorer и перейдите в папку Inetpub\wwwroot \FiveLakes\Config.
2. Создайте новый текстовый файл.
3. Введите следующий код:

```
Function mscsexecute(config, orderform, context, flags)
  Dim myMail
  Set FiveLakesMail = CreateObject("CDONTS.NewMail")
  FiveLakesMail.From = "Service@FiveLakes.com"
  FiveLakesMail.To = orderForm.ship_to_email
  FiveLakesMail.Subject = "Your order has been confirmed. Thank you."
  FiveLakesMail.Body = "This is a sample message."
  FiveLakesMail.Send
  Set FiveLakesMail = Nothing
End function
```

4. Сохраните файл как email.vbs, а затем закройте его.

Для того чтобы вставить компонент `Scriptor` в конвейер закупок (Purchase pipeline):

1. Нажмите кнопку **Start**, а затем укажите пункт **Programs**.
2. Укажите последовательно команды **Microsoft Site Server**, **Commerce**, а затем щелкните **Pipeline Editor**.
3. На экране появится окно **Commerce Server Pipeline Editor**. На панели инструментов нажмите кнопку **Open**.
4. В диалоговом окне **Open** перейдите в каталог Inetpub\wwwroot\FiveLakes \Config.
5. Выберите файл purchase.pcf, а затем нажмите кнопку **Open**.
6. Если конвейер не раскроется, то дважды щелкните кнопкой мыши на элементе **Accept**, чтобы он раскрылся.
7. Щелкните правой кнопкой мыши на элементе **SQLItemAdo**, а затем укажите команду **Insert Component**.
8. Щелкните **After**.
9. В диалоговом окне **Choose a Component** выберите компонент `Scriptor`, а затем щелкните по кнопке **OK**.
10. В конвейере дважды щелкните по объекту `Scriptor`.
11. В диалоговом окне **Component Properties** выберите опцию **External**.
12. Появится окно сообщений объекта `Scriptor` с вопросом, не хотите ли вы экспортировать внутренний исходный код. Щелкните по кнопке **No**.
13. Нажмите кнопку **Browse**.
14. В диалоговом окне **Open** выберите имя файла email.vbs, а затем нажмите кнопку **Open**.
15. Нажмите кнопку **OK**.
16. Перезагрузите сайт в Microsoft Management Console.

Вопросы для повторения

1. При смене папки сайта, где должна быть расположена новая папка?
2. Вы создали сайт на основе образца сайта, предоставленного в Microsoft Site Server Commerce Edition 3.0. Можете ли вы использовать Site Builder для настройки атрибутов продуктов и вариантов продуктов на сайте?
3. Какова функция объекта `NewMail`?

ПРИЛОЖЕНИЕ А

Файлы лабораторных работ

Предлагаемые файлы представляют собой завершенный программный код для всех лабораторных работ данного курса. Они включены в книгу, чтобы служить вам удобным примером при написании собственных программ.

Глава 1

default.asp

```
<html>
<body onload="onload() leftmargin="2" topmargin="2"
bgcolor="#FFFFFF" >
   <center>
      <H1>Ramona Publishing</H1>
   </center>
<center>
<form name="frmsignon" METHOD="POST" ACTION="validate.asp">
     <table>
        <tr>
           <td>Name:</td>
           <td><input type="text" name="login_id" size="20"></td>
        </tr>
        <tr>
           <td>Password:</td>
           <td><input type="password" name="password" size="20"></td>
        </tr>
     </table>
     <a href="javascript:signbutton()>
       <input type="Submit" Value="Sign On" name="Submit"></a>
     <table>
        <tr>
```

```
            <td><a href="newuser.asp">Become an e-Partner</a></td>
          </tr>
        </table>
      </form>
      </center>
      <Script Language="JavaScript">
         function onload()
         {
            document.frmsignon.login_id.focus();
         }
         function signbutton()
         {
           document.frmsignon.submit();
         }
      </Script>
</body>
</html>
```

newuser.asp

```
<html>
<body>
   <center><H1>Ramona Publishing</H1></center>
   <FORM METHOD="POST" ACTION="register.asp">
   <center><h3>e-Partner Registration</h3></center>
   <center><table>
     <tr>
        <td>Login ID</td>
        <td><input name="Lid" size="25"></td>
     </tr>
     <tr>
        <td>Password</td>
        <td><input name="pwd" size="25" type="password"></td>
     </tr>
     <tr>
        <td>Last Name</td>
        <td><input name="LName" size="25"></td>
     </tr>
     <tr>
        <td>First Name</td>
        <td><input name="FName" size="25">  </td>
     </tr>
     <tr>
        <td>Company</td>
        <td><input name="CName" size="25"></td>
     </tr>
```

```
      <tr>
         <td>Billing Contact</td>
         <td><input name="Billc" size="25"></td>
      </tr>
      <tr>
         <td>Address</td>
         <td><input name="Address" size="25"></td>
      </tr>
      <tr>
         <td>City</td>
         <td><input name="City" size="25"></td>
      </tr>
      <tr>
         <td>State/Province</td>
         <td><input name="State" size="25"></td>
      </tr>
      <tr>
         <td>Zip</td>
         <td><input name="Zip" size="25"></td>
      </tr>
      <tr>
         <td>Country</td>
         <td><input name="Country" size="25"></td>
      </tr>
      <tr>
         <td>Phone</td>
         <td><input name="Phone" size="25"></td>
      </tr>
      <tr>
         <td>Email</td>
         <td><input name="email" size="25"></td>
      </tr>
   </table></center>
   <center><input type="submit" value="Submit">
            <input type="reset" value="Reset"></center>
   </FORM>
</body>
</html>
```

home.asp

```
<%@ LANGUAGE=vbscript enablesessionstate=false LCID=1033 %>
<!—#INCLUDE FILE="i_shop.asp" —>
<HTML>
<HEAD>
```

```
  <TITLE><%= displayName %>: Lobby</TITLE>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset=ISO-8859-1">
</HEAD>
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#FF0000"
 VLINK="#FF0000" ALINK="#FF0000">
<!-#INCLUDE FILE="i_header.asp" ->
<H1><%= displayName %></H1>
<P>
Welcome to our shop. We have a broad range of products you
can choose from.
<%
sqlText = MSCSQueryMap.depts.SQLCommand
Set rsDepts = MSCS.Execute (sqlText, nDepts, adCmdText)
if rsDepts.EOF then
%>
<P>
There are currently no departments available.
<% else %>
<P>
Select a department below:
<UL>
<%
set dept_idField = rsDepts("dept_id")
set dept_nameField = rsDepts("dept_name")
do while Not rsDepts.EOF
%>
<LI><A HREF="<%= baseURL("dept.asp") &
mscsPage.URLShopperArgs("dept_id", dept_idField.value)
%>"><%= dept_nameField.value %></A>
<% rsDepts.MoveNext
loop
rsDepts.Close
MSCS.Close
%>
</UL>
<% end if %>
<BR>
You can see your order history on the <A HREF="<%=
pageURL("receipts.asp") %>">Order History Page</A>.
<!-#INCLUDE FILE="i_footer.asp" ->
</BODY>
</HTML>
```

register.asp

```
<%
   set con=server.createobject("adodb.connection")
   set rs=server.createobject("adodb.recordset")
   con.open "database=ramona;dsn=ramona;uid=sa;password=;"
   rs.open "ramona_epartner",con,2,2

   rs.addnew
      rs("Login_id")=request("lid")
      rs("Password")=request("pwd")
      rs("Lastname")=request("Lname")
      rs("firstname")=request("Fname")
      rs("Company_Name")=request("cname")
      rs("billing_contact")=request("billc")
      rs("Address")=request("address")
      rs("city")=request("city")
      rs("state")=request("state")
      rs("zip")=request("zip")
      rs("country")=request("country")
      rs("phone")=request("phone")
      rs("email")=request("email")
      rs("partner")=0
   rs.update
   rs.close
   con.close
   response.redirect "new_reg.asp"
%>
```

new_reg.asp

```
<html>
<body>
   <center><H1>Ramona Publishing</H1></center>
   <br><br><br>

   <font color="#ff00ff" face="verdana, arial, helvetica">
   <p align="center">
   <small>
      <strong>Thank you for signing up with Ramona Publishing.</strong>
  </small>
  </font>

   <font color="#ff00ff" face="verdana, arial, helvetica">
   <p align="center">
   <small>
```

```
      <strong>Your registration information has been recorded
              with the Support department.</strong>
   </small>
   </font>

   <font color="#ff00ff" face="verdana, arial, helvetica">
   <p align="center">
   <small>
      <strong>We will contact you shortly to finalize the set
              up of your account.</strong>
   </small>
   </font>
</body>
</html>
```

validate.asp

```
<%
   set con=server.createobject("adodb.connection")
   set rs=server.createobject("adodb.recordset")
   con.open "database=ramona;dsn=ramona;uid=sa;password=;"

   user=request.form("login_id")
   pwd=request.form("password")
   sqlstr= "select login_id, password, partner from "& _
   "ramona_epartner where login_id = '" & user & "' " &_
   " and (password = '" & pwd & "'" & " and partner = 1)
   set rs=con.execute(sqlstr)

   if rs.eof then
      response.redirect "default.asp"
   else
      response.redirect "home.asp"
   end if
%>
```

add_partner.asp

```
<%
set con=server.createobject("adodb.connection")
set rs=server.createobject("adodb.recordset")
con.open "database=ramona;dsn=ramona;uid=sa;password=;"
company=request.querystring("name")
sqlst= "select * from ramona_epartner Where company_name =
'" & company & "'"
rs.cursorlocation=3
rs.open sqlst,con,2,2
```

```
rs("partner")=1
rs.update
rs.close
con.close
response.redirect "e_info.asp"
%>
```

details.asp

```
<%@ LANGUAGE=vbscript enablesessionstate=false LCID=1033 %>
<!-#INCLUDE FILE="include/manager.asp" ->
<% REM  заголовок: %>
<% pageTitle = "e-Partner Manager" %>
<HTML>
<HEAD>
  <TITLE> <% = pageTitle %> </TITLE>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=ISO-8859-1">
   <!-#INCLUDE FILE="include/mgmt_define.asp" ->
</HEAD>

<BODY TOPMARGIN="8" LEFTMARGIN="8" BGCOLOR="#FFFFFF"
TEXT="#000000" LINK="#FF0000" ALINK="#FF0000"
VLINK="#FF0000">
<!-#INCLUDE FILE="include/mgmt_header.asp" ->
<%
set con=server.createobject("adodb.connection")
set rs=server.createobject("adodb.recordset")
con.open "database=ramona;dsn=ramona;uid=sa;password=;"
company=request.querystring("name")
sqlst= "select * from ramona_epartner Where company_name =
'" & company & "'"
set rs=con.execute(sqlst)
%>
<table border=1 width="700">
<caption><font COLOR="#000080" SIZE="4" FACE="Arial"><b>
Prospective e-Partner Information<b></font></caption>
   <tr>
      <td> Login ID <td>
      <td> <% =rs("login_id") %> </td>
   </tr>
   <tr>
      <td> Pas$word <td>
      <td> <% =rs("password") %> </td>
   </tr>
   <tr>
```

```
      <td> Last Name <td>
      <td> <% =rs("lastname") %> </td>
   </tr>
   <tr>
      <td> First Name <td>
      <td> <% =rs("firstname") %> </td>
   </tr>
   <tr>
      <td> Company <td>
      <td> <% =rs("Company_Name") %> </td>
   </tr>
   <tr>
      <td> Billing Contact <td>
      <td> <% =rs("billing_contact") %> </td>
   </tr>
   <tr>
      <td> Address <td>
      <td> <% =rs("Address") %> </td>
   </tr>
   <tr>
      <td> City <td>
      <td> <% =rs("city") %> </td>
   </tr>
   <tr>
      <td> State <td>
      <td> <% =rs("State") %> </td>
   </tr>
   <tr>
     <td> ZIP <td>
     <td> <% =rs("Zip") %> </td>
   </tr>
   <tr>
      <td> Country <td>
      <td> <% =rs("Country") %> </td>
   </tr>
   <tr>
      <td> Phone <td>
      <td> <% =rs("Phone") %> </td>
   </tr>
   <tr>
      <td> E-Mail <td>
      <td> <% =rs("email") %> </td>
   </tr>

</table>
<% if rs("partner")= 0 then %>
```

```
<A href="add_partner.asp?name=<% =rs("company_name")%> " >
Add to e-Partner List</a>
<% end if %>
```

e_info.asp

```
<%@ LANGUAGE=vbscript enablesessionstate=false LCID=1033 %>
<!-#INCLUDE FILE="include/manager.asp" ->
<% REM  заголовок: %>
<% pageTitle = "e-Partner Manager" %>
<HTML>
<HEAD>
  <TITLE> <% = pageTitle %> </TITLE>
  <META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=ISO-8859-1">
  <!-#INCLUDE FILE="include/mgmt_define.asp" ->
</HEAD>

<BODY TOPMARGIN="8" LEFTMARGIN="8" BGCOLOR="#FFFFFF" TEXT="#000000"
LINK="#FF0000" ALINK="#FF0000" VLINK="#FF0000">
<!-#INCLUDE FILE="include/mgmt_header.asp" ->
<%
set con=server.createobject("adodb.connection")
set rs=server.createobject("adodb.recordset")
con.open "database=ramona;dsn=ramona;uid=sa;password=;"
sqlst= "select Company_name, FirstName, LastName,
partner from ramona_epartner"
set rs=con.execute(sqlst)
%>
<html>
<body>
<table border=1 width="700">
  <caption><font COLOR="#000080" SIZE="4" FACE="Arial"><b>
  e-Partners<b></font> </caption>
    <tr>
      <th colspan =3> Existing e-Partners</th>
    </tr>
    <tr>
      <th> Company</th>
      <th> Last Name</th>
      <th> First Name</th>
    </tr>
    <%
    if not rs.eof then
    do until rs.eof
    if rs("partner") <> 0 then
    %>
```

```
    <tr>
      <td>
        <a href="details.asp?name=<%=rs("company_name")%>">
        <% =rs("Company_Name") %></a> </td>
           <td> <% =rs("LastName") %> </td>
           <td> <% =rs("FirstName") %> </td>
      </tr>
      <%
    end if
    rs.movenext
    loop
    End if
    %>
  </table>
<br><br><br><br>
<table border=1 width="700">
  <tr>
    <th colspan =3> Prospective e-Partners</th>
  </tr>
  <tr>
    <th> Company</th>
    <th> Last Name</th>
    <th> First Name</th>
  </tr>
  <%
  set rs=con.execute(sqlst)
  if not rs.eof then
  do until rs.eof
  if rs("partner") = 0 then
  %>
  <tr>
    <td>
    <a href="details.asp?name=<%=rs("company_name")%>">
    <% =rs("Company_Name") %></a> </td>
    <td> <% =rs("LastName") %> </td>
    <td> <% =rs("FirstName") %> </td>
  </tr>
  <%
  end if
  rs.movenext
  loop
  End if
  %>
  </table>
</body>
</html>
```

Глава 2

search.htm

```
<!- Copyright 1997,1998 Microsoft Corporation. All rights reserved. ->
<html>
<head>
  <title>Microsoft Site Server Search : Database Search</title>
  <meta http-equiv=content-type content="text/html;charset=iso-8859-1">
  <meta http-equiv="content-language" content="EN">
</head>

<body bgcolor="ffffff" leftmargin=15>
  <font face="verdana,arial,sans-serif" style="font-size:10pt">
  <img src="images/searchtitle.gif" hspace=0 vspace=2
       border=0 alt="Microsoft Site Server Search">
  <font face="verdana,arial,sans-serif" style="font-size:
        14pt; font-weight: bold">
    <p>Database Search:Ramona</p>
  </font>
  <form action="results.asp" method="get">
    <table cellpadding=0>
      <tr>
        <td><font face="verdana,arial,sans-serif"
                  style="font-size: 10pt">
            Enter text to search for: </font></td>
        <td><input type="text" name="qu" size="60"
             maxlength="100" value="">
        </td>
      </tr>
      <tr>
        <td><font face="verdana,arial,sans-serif"
                  style="font-size: 10pt">
            Special search for: </font></td>
        <td><select name=ss>
              <option value="">
              <option value="new">New Books
            </select>
        </td>
      </tr>
      <tr>
        <td> </td>
        <td align="right">
          <input type=Image align=bottom name="Search"
                 src="Images/SearchButton.gif" border=0>
        </td>
```

```
      <tr>
    </table>
  </form>
  <font color="#000000" face="Verdana,Arial,sans-serif"
        style="font-size: 10pt;">
   To find information in this database catalog, you can search
   the default field or other fields listed below.
  <ul>
    <li>
      To search the default field,
      type your query and then click <b>Search</b>.<p>
      <li>
        To search a different field, type the name of
        the field before the query, and then click
        <b>Search</b>.
        <br>
        <p>
        Example — type <b>@Author Smith</b> to find
        the word <i>Smith</i> in the Author field of the database.
  </ul>
  <p>For search syntax information, including how to
     search number or date fields, see <a href="dbtips.htm">
     Search Tips</a>.
  <p>
  </font>
  <b>Default field:</b> description<p>
  <b>Other searchable fields:</b>

  <ul>
    <li>sku
    <li>sale_price
    <li>sale_start
    <li>sale_end
    <li>author
    <li>edition
    <li>category
    <li>new
  </ul>
  </font>
</body>
</html>
```

results.asp

```
<%
' Results.asp: database search result page
' Страница результатов поиска в базе данных
```

```
' Copyright 1997,1998 Microsoft Corporation. All rights reserved.
%>
<html>
<head>
  <% L_ResultsPageTitle_Text = "Microsoft Site Server Search
     : Database Search"%>
  <title>
    <% = L_ResultsPageTitle_Text %>
  </title>
  <meta http-equiv=content-type content="text/html; charset=iso-8859-1">
  <meta http-equiv="content-language" content="EN">
</head>

<body bgcolor="ffffff" leftmargin=15>
  <font face="verdana,arial,sans-serif" style="font-size:10pt"">
    <font color="#333399" style="font-size: 14pt">
      <% L_PageHeading_text="Search Results" %>
      <% = L_PageHeading_text %>
    </font>
    <hr>
    <%
    ' Read in and initialize non-query variables from
    ' information posted to page.
    ' (Считывание и инициализация незапрашиваемых переменных
    ' из информации, отправленной на страницу)
    if Request("RecordNum") = "" then
      RecordNum=1
    else
      RecordNum=Request("RecordNum")
    end if
    %>
    <%
    ' Set up introductory sentence.
    ' Установка вводного предложения
    L_SearchingFor_text = "Searching for"

    if Request("qu") <> "" then
      Response.write L_SearchingFor_text & " <b>" &
Request("qu") & "</b>. "
    end if
    if Request("ss") <> "" then
      Response.write L_SearchingFor_text & " <b>" & _
Request("ss") & " <b> " & "Books" & "</b>. "
    end if
    %>
    <%
    if Request.QueryString <> "" then
```

```
' Set query and utility objects, and define query object properties.
' Установить объекты query и utility, указать свойства объекта query.
  set util = Server.CreateObject("MSSearch.util")
  set Q = Server.CreateObject("MSSearch.Query")
  Q.SetQueryFromURL(Request.QueryString)
  if request("ss") <> "" and request("qu") = "" Then
    q.query="@meta_new= " & "new"
  end if
  if request("ss") <> "" and request("qu") <> "" Then
    q.query=q.query & " and @meta_new= " & "new"
  end if
  ' Language must be set for the localeID of the query, as  well as
  ' searching for values in the DetectedLanguage catalog column.
  ' В запросе для идентификатора localeID должно быть установлено
  ' значение Language, а также произведен поиск значений в столбце
  ' каталога DetectedLanguage.

  if Request("language") <> "" then
    LocaleID = Util.ISOToLocaleID(Request("language"))
    if Q.Query <> "" then
      if Request("language") = "EN" then
        Q.Query="(" & Q.Query & ") & "&@DetectedLanguage^a " &
        LocaleID
      else
        Q.Query="(" & Q.Query & ") & "&@DetectedLanguage =^s " &
        LocaleID
      end if
    else
      if Request("language") = "EN" then
        Q.Query="@DetectedLanguage ^a " & LocaleID
      else
        Q.Query="@DetectedLanguage = ^s " & LocaleID
      end if
    end if
    Q.LocaleId = LocaleID
  end if

  Q.MaxRecords = 25
  Q.SortBy = "Rank[d],DocTitle"

  ' The database wizard creates the following Q.Catalog,
  ' Q.DefineColumn, Q.Columns lines.
  ' Мастер базы данных создаст следующие строки: Q.Catalog,
  ' Q.DefineColumn, Q.Columns.

  Q.Catalog = "Ramona"
  Q.DefineColumn "sku = d1b5d3f0-c0b3-11cf-9a92-00a0c908dbf1 sku"
```

```
      Q.DefineColumn "list_price (DBTYPE_I4) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 list_price"
      Q.DefineColumn "image_file = d1b5d3f0-c0b3-11cf-9a92-
00a0c908dbf1 image_file"
      Q.DefineColumn "image_width (DBTYPE_I4) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 image_width"
      Q.DefineColumn "image_height (DBTYPE_I4) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 image_height"
      Q.DefineColumn "sale_price (DBTYPE_I4) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 sale_price"
      Q.DefineColumn "sale_start (VT_FILETIME) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 sale_start"
      Q.DefineColumn "sale_end (VT_FILETIME) = d1b5d3f0-c0b3-
11cf-9a92-00a0c908dbf1 sale_end"
      Q.DefineColumn "author = f29f85e0-4ff9-1068-ab91-08002b27b3d9 4"
      Q.DefineColumn "edition = d1b5d3f0-c0b3-11cf-9a92-
00a0c908dbf1 edition"
      Q.DefineColumn "category = d5cdd502-2e9c-101b-9397-08002b2cf9ae 2"
      Q.DefineColumn "new = d1b5d3f0-c0b3-11cf-9a92-00a0c908dbf1 new"
      Q.Columns = "DocTitle, DocAddress, Description,
sale_price, sale_start, sale_end, author, edition, category, new"

 ' Create the recordset holding the search results.
 ' Создать набор записей (recordset) для хранения результатов поиска.

  on error resume next
  set RS = Q.CreateRecordSet("sequential")
  if err then
    createerror = err.description
  end if

  ' Log query for reporting with the Site Server Analysis feature of
  ' Microsoft Site Server. Only log query for first page of
  ' search results.
  ' Зарегистрируйте результаты запроса,
  ' полученные  с помощью программы Site Server Analysis,
  ' входящей в состав Microsoft Site Server.
  ' Записывайте только первую страницу результатов запроса.

  if Q.StartHit = "" then
    InfoToLog = "&MSS.request.Search Catalog=" & Q.Catalog & _
    "&MSS.request.Search Start Hit=0"
    if Request("qu") <> "" then
      InfoToLog = InfoToLog & "&MSS.request.Search Query="& Request("qu")
    end if
    if err or (RS.EOF and RS.BOF) then
      InfoToLog = InfoToLog & "&MSS.request.Search RowCount=0"
```

```
    else
      InfoToLog = InfoToLog & "&MSS.request.Search RowCount=" & _
        RS.Properties("RowCount")
    end if
    ' Information to be appended cannot contain commas or spaces.
    ' Приписываемая в конец информация не должна содержать
    ' запятых или пробелов.

    InfoToLog = Replace(InfoToLog, " ", "%20")
    InfoToLog = Replace(InfoToLog,",","+")
    Response.AppendToLog InfoToLog
  end if
  %>
  <%
  ' If the query can't be executed, display the error description.
  ' Если запрос не может быть выполнен, то необходимо отобразить
  ' описание ошибки.

  if err then
    Response.write createerror
    L_Syntax_Error = "For information on search syntax, see
<a href=dbtips.htm>Search Tips</a>"
    Response.write "<br>" & L_Syntax_Error & "."

  ' If no matches are found, display a message.
  ' Если соответствий не найдено, то необходимо выдать сообщение.

  elseif RS.BOF and RS.EOF then
    if Q.QueryIncomplete=true then
    L_TooComplex_Error = "The query is too complex. Try _
      using a simpler query. <br>For information on search _
      syntax, see <a href=dbtips.htm>Search Tips</a>."
    Response.write L_TooComplex_Error
  else
    L_NoMatch_Error = "No records matching your query were found.
 <br>For suggestions on how to broaden your search, see
 <a href=dbtips.htm>Search Tips</a>."
    Response.write L_NoMatch_Error
  end if

  ' If query could be executed, and display table showing
  ' number of results found.
  ' Если запрос может быть выполнен,
  ' и в таблице представлено количество найденных результатов.
  else
    L_MoreThan500_text = "More than 500"
    L_Found_text = "Found"
```

```
    L_Showing_text = "Showing"
    ' Set up number found.
    ' Установить количество найденных результатов.
    if RS.Properties("RowCount") > 500 then
      NumberFound = L_MoreThan500_text
    else
      NumberFound= RS.Properties("RowCount")
    end if

  ' If more than one result is found, display which results are shown.
  ' Если найдено более одного результата,
  ' отобразить, какие результаты показаны.
  if RS.Properties("RowCount") <> 1 then
    ' If only one on any but first page of results.
    ' Если одна из всех, но первая страница результатов.
    if RS.Properties("RowCount") = cInt(RecordNum) then
      Displayed = RecordNum
      ' For all pages except last page.
      ' Для всех страниц, кроме последней
    elseif RS.Properties("RowCount") > (RecordNum +Q.MaxRecords -1) then
    Displayed = RecordNum & " - " & (RecordNum +Q.MaxRecords - 1)

    ' For last page with more than one result.
    ' Для последней страницы с более чем одним результатом.
    else
      Displayed = RecordNum & " - " & RS.Properties("RowCount")
    end if
  else
    Displayed = "1"
  end if
%>

<% ' outermost table %>
<% ' внешняя таблица %>

  <table>

<% ' table with number of results %>
<% ' таблица с количеством результатов %>

  <tr>
    <td>
      <table width=75% align=center border=1>
        <tr>
          <td align=center nowrap bgcolor=80BBDD>
          <fontstyle="font-size: 10pt">
          <% Response.write L_Found_text & ": " & NumberFound %>
          </font></td>
```

```
          <td align=center bgcolor=80BBDD nowrap>
          <fontstyle="font-size: 10pt">
          <% Response.write L_Showing_text & ": " & Displayed %>
          </font></td>
        </tr>
      </table>
    </td>
  </tr>

  <br>
<%
  ' Set up loop to iterate through results.
  ' Установка цикла с итерациями по результату.
  Do while not RS.EOF
   ' If document title is blank, use <No title>: file name instead.
   ' When visitors sort by title, all blank titles sort at the
   ' top of the list. By labeling the substitution, visitors will
   ' be less confused by the sorted results.
   ' Если заголовок документа пустой, то вместо имени файла
   ' используйте <No title>:.
   ' Когда посетители осуществляют сортировку
   ' по заголовку, все пустые заголовки
   ' размещаются в верхней части списка.
   ' Если их снабдить надписями, то посетители не будут смущены
   ' результатами сортировки.

    if RS("DocTitle") <> "" then
      Title = RS("DocTitle")
    else
      L_Untitled_text = "No title"
      Title = L_Untitled_text
    end if
%>

<% ' One table is used for each search result. %>
<% ' Для каждого результата поиска используется одна таблица. %>

  <tr>
    <td>
      <table cellpadding=0 cellspacing=0>
        <tr>
          <td width=21><font style="font-size: 10pt">
<%
  iRank = RS("Rank")
  If (Not IsNull(Rank)) then
    If (iRank > 750) then
      Response.Write("<img src=""Images/RankHighest.gif""
                     hspace=4 width=21 height=6 border=0>")
```

```
    elseif (iRank > 500) then
      Response.Write("<img src=""Images/RankHigh.gif""
                     hspace=4 width=21 height=6 border=0>")
    elseif (iRank > 250) then
      Response.Write("<img src=""Images/RankLow.gif""
                     hspace=4 width=21 height=6 border=0>")
    else
      Response.Write("<img src=""Images/RankLowest.gif""
                     hspace=4 width=21 height=6 border=0>")
    end if
  else
    Response.Write("<img src=""Images/RankNone.gif""
                   width=21 height=6 border=0>")
   end if
%>
     </font></td>
     <td bgcolor=80BBDD><font style="font-size: 10pt">
      <a href="<% = RS("DocAddress") %>" style="color=#000000">
        <% = Title %></a>
     </font></td>
    </tr>
    <tr>
    <td></td>
    <td><font style="font-size: 10pt">
      <% Response.write RS("Description") %>
     </font></td>
   </tr>
    <tr>
     <td></td>
     <td height=5></td>
    </tr>
   </table>
  </td>
 </tr>

<%
   ' Increment the results.
   ' Приращение результатов.

   RS.MoveNext
   RecordNum = RecordNum + 1
  Loop
%>

  <hr>

<%
  ' If there are more results pages, set up the "More Results" link.
```

```
  ' Если страниц результатов больше, установите соответствующую ссылку
  ' "More Results".

  if RS.Properties("MoreRows") = true then
    Q.StartHit = RS.Properties("NextStartHit")

    ' Repeat query with new start hit.
    ' Повторить запрос с новыми начальными значениями.
    L_MoreResults_link = "More Results"
    MoreLink = "<a href=results.asp?" & Q.QueryToURL & "&" _
      & "DisplayText=" & Server.URLEncode(DisplayText) & "&" _
      & "RecordNum=" & RecordNum & ">" & L_MoreResults_link & "</a>"
  end if
%>
 <tr>
  <td>
   <table width=75% align=center border=1>
    <tr>
     <td align=center bgcolor=80BBDD nowrap><fontstyle="font-size: 10pt">
      <% Response.write L_Found_text & ": " & NumberFound %>
      </font></td>
     <td align=center bgcolor=80BBDD nowrap><fontstyle="font-size: 10pt">
      <% Response.write L_Showing_text & ": " & Displayed %>
      </font></td>
     <% if RS.Properties("MoreRows") = true then %>
      <td align=center bgcolor=80BBDD nowrap>
        <font style="font-size:10pt">
      <% = MoreLink %>
      </font></td>
     <% end if %>
    </tr>
   </table>
  <br>
  </td>
 </tr>

<% ' end of outermost table %>
<% ' конец внешней таблицы %>

</table>

<%
 end if
end if
%>

</font>
</body>
</html>
```

Глава 3

i_util.asp

```
<%
function UtilGetOrderFormStorage()
  Set orderFormStorage = Server.CreateObject("Commerce.DBStorage")
  Call
  orderFormStorage.InitStorage(MSCSSite.DefaultConnectionString, _
    "fivelakes_basket", "shopper_id", "Commerce.OrderForm", _
    "marshalled_basket", "date_changed")
  Set UtilGetOrderFormStorage = orderFormStorage
end function

function UtilGetOrderForm(byRef orderFormStorage, byRef created)
  created = 0
  On Error Resume Next
  Set orderForm = orderFormStorage.GetData(null, mscsShopperID)
  On Error Goto 0
  if IsEmpty(orderForm) then
    set orderForm = Server.CreateObject("Commerce.OrderForm")
    orderForm.shopper_id = mscsShopperID
    created = 1
  end if
  Set UtilGetOrderForm = orderForm
end function

function UtilPutOrderForm(byRef orderFormStorage, byRef orderForm, _
                          byRef created)
  if created = 0 then
    Call orderFormStorage.CommitData(NULL, orderForm)
  else
    Call orderFormStorage.InsertData(NULL, orderForm)
  end if
end function

function UtilGetReceiptStorage()
  REM Create a storage object for receipts
  REM Создать объект для хранения квитанций
  Set receiptStorage =   Server.CreateObject("Commerce.DBStorage")
  Call
  receiptStorage.InitStorage(MSCSSite.DefaultConnectionString, _
    "fivelakes_receipt", "order_id", "Commerce.OrderForm", _
    "marshalled_receipt", "date_entered")
  receiptStorage.Mapping.Value("_total_total") = "total"
  Set UtilGetReceiptStorage = receiptStorage
end function
```

```
function UtilGetPipeContext()
  Set pipeContext =Server.CreateObject("Commerce.Dictionary")
  Set pipeContext("MessageManager") = MSCSMessageManager
  Set pipeContext("DataFunctions") = MSCSDataFunctions
  Set pipeContext("QueryMap") = MSCSQueryMap
  Set pipeContext("ConnectionStringMap") = MSCSSite.ConnectionStringMap
  pipeContext("SiteName") = displayName
  pipeContext("DefaultConnectionString")=MSCSSite.DefaultConnectionString
  pipeContext("Language") = "USA"
  Set UtilGetPipeContext = pipeContext
end function

function UtilRunPipe(file, orderForm, pipeContext)
  Set pipeline = Server.CreateObject("Commerce.MtsPipeline")
  Call
  pipeline.LoadPipe(Request.ServerVariables("APPL_PHYSICAL_PATH") & _
    "config\" & file)
  REM Call
  REM Вызов

  pipeline.SetLogFile(Request.ServerVariables("APPL_PHYSICAL_PATH") & _
    "config\pipeline.log")
  errorLevel = pipeline.Execute(1, orderForm, pipeContext, 0)
  UtilRunPipe = errorLevel
end function

function UtilRunTxPipe(file, orderForm, pipeContext)
  Set pipeline =
  Server.CreateObject("Commerce.MtsTxPipeline")
  Call
  pipeline.LoadPipe(Request.ServerVariables("APPL_PHYSICAL_PATH") & _
    "config\" & file)
  REM Call
  REM Вызов

  pipeline.SetLogFile(Request.ServerVariables("APPL_PHYSICAL_PATH") & _
    "config\txpipeline.log")
  errorLevel = pipeline.Execute(1, orderForm, pipeContext, 0)
  UtilRunTxPipe = errorLevel
end function

function UtilRunPlan()
  REM Create a storage object for the order forms
  REM Создание объекта для хранения форм заказов (order forms)

  Set mscsOrderFormStorage = UtilGetOrderFormStorage()
  REM Get the orderform
  REM Получение формы заказа (orderform)
```

```
  Set mscsOrderForm = UtilGetOrderForm(mscsOrderFormStorage, created)
  REM Get the basic pipe context
  REM Получение основного контекста конвейера

  Set mscsPipeContext = UtilGetPipeContext()
  REM Create and run the pipe
  REM Создание и запуск конвейера

  errorLevel = UtilRunPipe("CorpPlan.pcf",mscsOrderForm,mscsPipeContext)
  REM Save the order form in case running the pipe made
  REM changes to the order form
  REM Сохранение формы заказа, если в результате запуска конвейера
  REM в форму заказа были внесены изменения

  if created then
    Call mscsOrderFormStorage.InsertData(null, mscsOrderForm)
  else
    Call mscsOrderFormStorage.CommitData(null, mscsOrderForm)
  end if
  Set UtilRunPlan = mscsOrderForm
end function
%>
```

xt_orderform_purchase.asp

```
<%@ LANGUAGE=vbscript enablesessionstate=false LCID=1033 %>
<!-#INCLUDE FILE="i_shop.asp" ->
<!-#INCLUDE FILE="i_util.asp" ->
<%
function OrderFormPurchaseArgs(byRef orderForm, byRef errorList)
  REM - cc info:
  REM - информация cc:

  cc_name = mscsPage.RequestString("cc_name", null, 1, 100)
  if IsNull(cc_name) then
    errorList.Add("Credit card name must be a string between _
      1 and 100 characters")
  else
    orderForm.cc_name = cc_name
  end if
  cc_type = mscsPage.RequestString("cc_type", null, 1, 100)
  orderForm.cc_type = cc_type
  if IsNull(cc_type) then
    errorList.Add("Credit card type must be a string between _
    1 and 100 characters")
```

```
else
  orderForm.cc_type = cc_type
end if
cc_number = mscsPage.RequestString("_cc_number", null, 13, 19)
if IsNull(cc_number) then
  errorList.Add("Credit card number must be a string _
    between 13 and 19 characters")
else
 orderForm.[_cc_number] = cc_number
end if
cc_expmonth = mscsPage.RequestNumber("_cc_expmonth", null, 1, 12)
if IsNull(cc_expmonth) then
  errorList.Add("Expiration month must be a number between 1 and 12")
else
  orderForm.[_cc_expmonth] = cc_expmonth
end if
cc_expyear = mscsPage.RequestNumber("_cc_expyear", null, 1997, 2003)
if IsNull(cc_expyear) then
errorList.Add("Expiration year must be a number between 1997 and 2003")
else
  orderForm.[_cc_expyear] = cc_expyear
end if
REM - bill to:
REM - отослать счет к...:

bill_to_name = mscsPage.RequestString("bill_to_name", null, 1, 100)
if IsNull(bill_to_name) then
  errorList.Add("Ship to name must be a string between 1 and 100 _
    characters")
else
  orderForm.bill_to_name = bill_to_name
end if
bill_to_street = mscsPage.RequestString("bill_to_street", null, 1, 100)
if IsNull(bill_to_street) then
  errorList.Add("Ship to street must be a string between 1 and 100 _
    characters")
else
  orderForm.bill_to_street = bill_to_street
end if
bill_to_city = mscsPage.RequestString("bill_to_city", null, 1, 100)
if IsNull(bill_to_city) then
  errorList.Add("Ship to city must be a string between 1 and 100 _
    characters")
else
  orderForm.bill_to_city = bill_to_city
end if
```

```
  bill_to_state = mscsPage.RequestString("bill_to_state", null, 1, 100)
  if IsNull(bill_to_state) then
    errorList.Add("Ship to zip must be a string between 1 and 100 _
      characters")
  else
    orderForm.bill_to_state = bill_to_state
  end if
  bill_to_zip = mscsPage.RequestString("bill_to_zip", null, 1, 100)
  if IsNull(bill_to_zip) then
    errorList.Add("Ship to zip must be a string between 1 and 100 _
      characters")
  else
   orderForm.bill_to_zip = bill_to_zip
  end if
  bill_to_country = mscsPage.RequestString("bill_to_country",null,1,100)
  if IsNull(bill_to_country) then
    errorList.Add("Ship to country must be a string between 1 and 100 _
      characters")
  else
    orderForm.bill_to_country = bill_to_country
  end if
  bill_to_phone = mscsPage.RequestString("bill_to_phone", null, 1, 100)
  if IsNull(bill_to_phone) then
    errorList.Add("Ship to phone must be a string between 1 and 100 _
      characters")
  else
    orderForm.bill_to_phone = bill_to_phone
  end if
  OrderFormPurchaseArgs = true
end function

function OrderFormPurchase(byRef errorList)
  OrderFormPurchase = null
  REM Create a storage object for the order form
  REM Создать объект для хранения формы заказа

  Set mscsOrderFormStorage = UtilGetOrderFormStorage()
  REM Retrieve order from the storage
  REM Извлечь форму заказа из объекта хранения
  Set mscsOrderForm = UtilGetOrderForm(mscsOrderFormStorage, created)
  REM Retrieve args from form:
  REM Извлечь аргументы из формы
  Call OrderFormPurchaseArgs(mscsOrderForm, errorList)
  REM If the order form has no items, add an error
  REM Если в форме заказа нет элементов, добавить сообщение об ошибке
```

```
if mscsOrderForm.Items.Count = 0 then
  errorList.Add("No items to order.")
end if
if errorList.Count > 0 then
  REM Save changes to the order form so far
  REM Сохранить изменения в форме заказа на данный момент времени
  call UtilPutOrderForm(mscsOrderFormStorage, mscsOrderForm, created)
  exit function
end if
REM Set the verify with flags onto the orderform
REM Установить проверку в соответствии с флагами на форме заказа

call mscsPage.ProcessVerifyWith(mscsOrderForm)
REM Create the basic pipe context
REM Создать основной контекст конвейера
set mscsPipeContext = UtilGetPipeContext()
REM Run the plan
REM Запустить план

errorLevel = UtilRunPipe("CorpPlan.pcf",mscsOrderForm,mscsPipeContext)
REM — Finally if no errors, run the actual purchase
REM — Create a transacted pipeline for this execution
REM — Наконец, если нет ошибок, запустить реальную закупку
REM — Для этого создать конвейер транзакции
if mscsOrderForm.[_Basket_Errors].Count = 0 and _
  mscsOrderForm.[_Purchase_Errors].Count = 0 and errorLevel = 1 then
  REM Create the receipt storage
  REM Создать память для квитанции
  Set mscsReceiptStorage = UtilGetReceiptStorage()
  REM Add the receipt storage into the pipe context...the
  Save Receipt component uses it
  REM Включить память квитанции в контекст конвейера
  Set mscsPipeContext.ReceiptStorage = MSCSReceiptStorage
  REM Run the transacted pipe
  REM Запустить конвейер транзакции
  errorLevel = UtilRunTxPipe("purchase.pcf",
  mscsOrderForm, mscsPipeContext)
  errorLevel = UtilRunTxPipe("CorpSubmit.pcf",
  mscsOrderForm, mscsPipeContext)
end if
if mscsOrderForm.[_Basket_Errors].Count > 0 then
  REM — goto basket to show errors
  REM перейти на страницу basket для отображения ошибок
  Response.redirect "basket.asp?" & mscsPage.URLShopperArgs()
  Response.End
end if
```

```
  if mscsOrderForm.[_Purchase_Errors].Count > 0 or errorLevel > 1 then
    if mscsOrderForm.[_Purchase_Errors].Count > 0 then
      for each errorStr in mscsOrderForm.[_Purchase_Errors]
        errorList.Add(errorStr)
      next
    else
      errorList.Add("Unable to complete purchase at this time")
    end if
    OrderFormPurchase = null
    exit function
  end if
  REM Save the order id before we delete it
  REM Сохранение идентификатора заказа (order_id) перед его удалением
  order_id = mscsOrderForm.order_id
  REM Purchase was successful....delete the order form from the storage
  REM Заказ на закупку выполнен успешно....удаление формы
  REM заказа из памяти

  call MSCSOrderFormStorage.DeleteData(null, mscsOrderForm)
  REM Return the order id
  REM Возврат идентификатора заказа
  OrderFormPurchase = order_id
end function

Set errorList = Server.CreateObject("Commerce.SimpleList")
order_id = OrderFormPurchase(errorList)
if errorList.Count > 0 then
%>
<!-#INCLUDE FILE="i_error.asp" ->
<%
else
call Response.Redirect("confirmed.asp?" &
mscsPage.URLShopperArgs("order_id", order_id))
end if
%>
```

potemplate.txt

```
<%% Dim nItemcount
  nItemCount = Items.Items.count
  if (nItemCount > 0) then
   dim iItem,Item
   for iItem = 0 to nItemCount - 1
   Set Item = Items.Items.Item(iItem)
%%>
```

```
  Order ID : <%% =items.order_id %%>
  Item Number:<%%= item.sku %%>
  Qty:<%%= item.quantity %%>
  Total Cost:<%%= items.[_total_total] %%>
<%% next end if %%>
```

calltransmit.vbs

```
Function mscsexecute(config, orderform, context, flags)
  mscsexecute = 1
  Dim mscsMtsTxPipeline
  Set mscsMtsTxPipeline = CreateObject("Commerce.MtsTxPipeline")
  call
  mscsMtsTxPipeline.loadpipe("c:\inetpub\wwwroot\FiveLakes\ _
                              config\transmit.pcf")
  Dim TransportDictionary
  Set TransportDictionary = CreateObject("Commerce.Dictionary")

  Set TransportDictionary.object = orderform
  TransportDictionary.Receipt_requested = "yes"

  errorlevel = mscsMtsTxPipeline.Execute(1, TransportDictionary, _
    context, 0)
End Function
```

Глава 4

getdata.asp

```
<%
  Dim TransportDictionary
  set TransportDictionary = server.createobject("commerce.dictionary")
  TransportDictionary.working_data = request.form

  'Create a Pipeline Object, load a PCF
  'Создание объекта конвейера, загрузка PCF-файла
  Set pipe = Server.CreateObject("Commerce.MtsTxPipeline")
  call pipe.loadpipe("c:\inetpub\wwwroot\ramona\config\receive.pcf")

  'Create a context dictionary and execute
  'Создание и выполнение словаря контекста
  Set context = Server.CreateObject("Commerce.Dictionary")
  errorlevel = pipe.Execute(1, TransportDictionary, context, 0)
%>
```

poaccept.vbs

```
Function mscsexecute(config, transportdictionary, context, flags)
  Dim Connection
  Set Connection = CreateObject("ADODB.Connection")
  Connection.provider="sqloledb"
  Connection.Open "datasource=ramona; initial _
    catalog=ramona; user id=sa; password=;"

  Dim success
  success = VendorInsertitems(transportdictionary.working_data, _
            Connection,Context)
  If success Then
    mscsexecute = 1
  Else
    mscsexecute = 3
  End If
  Connection.close
End Function
Function VendorInsertitems(ByRef working_data, ByRef Connection, _
                           ByRef Context)
  VendorInsertitems = True
  dim rs, po_no
  set rs=createobject("adodb.recordset")
  rs.open "ramona_po",connection,2,2
  if rs.eof then
    po_no = 1
  else
    rs.movelast
    po_no = rs("po_no")
    po_no = po_no + 1
  end if
  For Each item In working_data.items
    rs.addnew
    rs("po_no")=po_no
    rs("sku")=item.sku
    rs("name")=item.name
    rs("quantity")=item.quantity
    rs.update
  Next
  rs.close
  If (0 < Connection.Errors.Count) then
    VendorInsertitems = False
  End If
End Function
```

accreceipt.asp

```
<%
  Dim TransportDictionary
  set TransportDictionary = server.createobject("commerce.dictionary")
  TransportDictionary.working_data = request.form

  'Create a Pipeline Object, load a PCF
  'Создание объекта конвейера, загрузка PCF-файла

  Set pipe = Server.CreateObject("Commerce.MtsTxPipeline")
  call pipe.loadpipe("c:\inetpub\wwwroot\fivelakes\config\receive.pcf")

  'Create a context dictionary and execute
  'Создание и выполнение словаря контекста

  Set context = Server.CreateObject("Commerce.Dictionary")
  errorlevel = pipe.Execute(1, TransportDictionary, context, 0)
%>
```

Глава 5

cipmreceive.vbp (cipmrcv.cls)

```
Option Explicit
Implements IPipelineComponent
Implements ISpecifyPipelineComponentUI
Implements IPipelineComponentDescription
Implements IPipelineComponentAdmin
Implements IPersistDictionary
Dim strFilename As String
Dim fDirty As Boolean

Private Sub IPipelineComponent_EnableDesign(ByVal fEnable As Long)
End Sub

Private Function IPipelineComponent_Execute(ByVal dictTransport As _
  Object, ByVal pdispContext As Object, ByVal lFlags As Long) As Long
  Dim numItems
  Dim i
  Dim BookOrder
  Dim item
  Set BookOrder = dictTransport.object
  On Error GoTo Error
  If strFilename = "" Then
    GoTo Error
  End If
```

```
  Open strFilename For Output As #1
  Write #1, "Customer Details"
  Write #1, BookOrder.Customer
  Write #1, BookOrder.Address
  Write #1, BookOrder.City & ", " & BookOrder.State & " " & BookOrder.Zip
  Write #1, "BookOrder Details"
  Write #1, "Number of items =" & Str(BookOrder.Items.Count)
  For Each item In BookOrder.Items
    Write #1, item.ISBN & ", " & item.Title
  Next
  Close #1
  IPipelineComponent_Execute = 1
  Exit Sub
Error:
  IPipelineComponent_Execute = 3
End Function

Private Function _
ISpecifyPipelineComponentUI_GetPipelineComponentUIProgID() As String
  ISpecifyPipelineComponentUI_GetPipelineComponentUIProgID = _
    "CIPMReceiveUI.ComponentUI"
End Function

Private Function IPipelineComponentDescription_ContextValuesRead() As _
        Variant
  IPipelineComponentDescription_ContextValuesRead =Array("None")
End Function

Private Function IPipelineComponentDescription_ValuesRead() As Variant
  IPipelineComponentDescription_ValuesRead = Array("working_data")
End Function

Private Function IPipelineComponentDescription_ValuesWritten() As Variant
  IPipelineComponentDescription_ValuesWritten = Array("None")
End Function

Private Function IPipelineComponentAdmin_GetConfigData() As Object
  Dim objectConfig As New CDictionary
  objectConfig.Value("filename") = strFilename
  Set IPipelineComponentAdmin_GetConfigData = objectConfig
End Function

Private Sub IPipelineComponentAdmin_SetConfigData(ByVal pDict As Object)
  strFilename = pDict.Value("filename")
End Sub

Private Function IPersistDictionary_GetProgID() As String
  IPersistDictionary_GetProgID = "CIPMReceive.CIPMRcv"
End Function
```

```
Private Sub IPersistDictionary_InitNew()
  strFilename = ""
  fDirty = False
End Sub

Private Function IPersistDictionary_IsDirty() As Long
  IPersistDictionary_IsDirty = fDirty
End Function

Private Sub IPersistDictionary_Load(ByVal pdispDict As Object)
  strFilename = pdispDict.Filenm
End Sub

Private Sub IPersistDictionary_Save(ByVal pdispDict As Object, _
        ByVal fSameAsLoad As Long)
  pdispDict.Filenm = strFilename
End Sub
```

Глава 6

book.dtd

```
<!ELEMENT book_description (Book_Title, Author+, Edition?,
Price, Cover_image?, About_the_Book*)>
  <!ELEMENT Book_Title (#PCDATA)>
  <!ELEMENT Author (#PCDATA)>
  <!ELEMENT Edition (#PCDATA)>
  <!ELEMENT Price (#PCDATA)>
  <!ELEMENT Cover_image (#PCDATA)>
  <!ELEMENT About_the_Book (#PCDATA)>
```

book.xml

```
<?xml version="1.0" ?>
  <!DOCTYPE book_description SYSTEM "book.dtd">
  <book_description>
<Book_Title>Mastering Windows NT Server 4</Book_Title>
   <Author>Mark Minasi</Author>
   <Author>Peter Dyson</Author>
   <Edition>Oct 1997</Edition>
   <Price>41.99</Price>
<About_the_Book>A high-level, irreverent, yet
readable explanation of NT Server in the Enterprise
provides the best discussion of NT architecture and
TCP/IP in print, and includes scores of undocumented
secrets, tips, tricks, and workarounds.
   </About_the_Book>
  </book_description>
```

Глава 7

book.xml

```
<?xml version="1.0"?>
<!DOCTYPE book_description SYSTEM "book.dtd">
<?xml-stylesheet type="text/xsl" href="book.xsl" ?>
<book_description>
  <book_title>Computer Networks and Internet</book_title>
  <author>Douglas E. Comer</author>
  <author>Ralph E. Droms</author>
  <publisher>Prentice Hall</publisher>
  <edition>October 1996</edition>
  <price>77.25</price>
  <review written_by="Publisher">
This book provides a thorough state-of-the-art overview
of networking and Internet technology and is ideal for
those with little or no background in the subject.
  </review>
</book_description>
```

book.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
  <HTML>
    <HEAD>
<TITLE>Five Lakes Publishing: Product: 'Computer
Networks and Internet</TITLE>
    </HEAD>
<BODY BGCOLOR="#FFFFFF" WIDTH="640" TOPMARGIN="2" LEFTMARGIN="2">
<TABLE BORDER="0" WIDTH="613" HEIGHT="40"
CELLSPACING="0" CELLPADDING="0" BGCOLOR="#00FFFF">
      <TR>
      <TH WIDTH="613" COLSPAN="2" HEIGHT="31"
VALIGN="Baseline" BGCOLOR="#FFFFFF">
         <P Align="LEFT">
<IMG SRC="images/Top.gif" alt="Top.gif" WIDTH="614" HEIGHT="50"/>
         </P>
      </TH>
      </TR>
</TABLE>

<TABLE BORDER="0" CELLPADDING="0"
CELLSPACING="0" ALIGN="LEFT">
```

```
      <TR>
        <TD WIDTH="1" ROWSPAN="2"
BGCOLOR="#9ED7F8" VALIGN="top" ALIGN="left"><A
href="http://irdsiteserver:80/fivelakes/default.asp?mscssid
=WP1LHW5U6US12KFN00GG9VSGDXNH72A5"><img
src="images/home1.gif" width="90" HEIGHT="51" BORDER="0"
ALT="Home" ALIGN="TOP" /></A><BR/>
          <A
HREF="http://irdsiteserver:80/fivelakes/basket.asp?mscssid=
NW1LHW5U6US12KFN00GG9VSGDPPQ3GUC"><IMG
SRC="images/cart.gif" WIDTH="90" HEIGHT="48" BORDER="0"
ALT="Cart" ALIGN="TOP"/></A><BR/>
          <A
HREF="http://irdsiteserver:80/fivelakes/shipping.asp?mscssi
d=NW1LHW5U6US12KFN00GG9VSGDPPQ3GUC"><IMG
SRC="images/checkout.gif" WIDTH="90" HEIGHT="48" BORDER="0"
ALT="Checkout" ALIGN="TOP"/></A><BR/>
          <A
HREF="http://irdsiteserver:80/fivelakes/ordertrack.asp?mscs
sid=NW1LHW5U6US12KFN00GG9VSGDPPQ3GUC"><IMG
SRC="images/order.gif" WIDTH="90" HEIGHT="47" BORDER="0"
ALT="OrderStatus" ALIGN="TOP"/></A><BR/>
          <A
HREF="http://irdsiteserver:80/fivelakes/find.asp?mscssid=NW
1LHW5U6US12KFN00GG9VSGDPPQ3GUC"><IMG
SRC="images/search.gif" WIDTH="90" HEIGHT="46" BORDER="0"
ALT="Search" ALIGN="TOP"/></A><BR/>
          <A
HREF="http://irdsiteserver:80/fivelakes/faq.asp?mscssid=NW1
LHW5U6US12KFN00GG9VSGDPPQ3GUC"><IMG SRC="images/faq.gif"
WIDTH="90" HEIGHT="62" BORDER="0" ALT="FAQ"
ALIGN="TOP"/></A></TD>
      </TR>
        <TD WIDTH="100" HEIGHT="680" VALIGN="Top" BGCOLOR="#FFFFFF"></TD>
    </TABLE>

    <TABLE WIDTH="400" HEIGHT="80" BGCOLOR="#FFFFFF">
      <TR>
        <TD>
          <P ALIGN="Center">
            <FONT FACE="Arial" SIZE="2">
              <strong>Product Information</strong>
            </FONT>
          </P>
        </TD>
      </TR>
    </TABLE>
```

```
    <TABLE WIDTH="400">
     <xsl:apply-templates select="book_description/book_title" />
     <tr>
        <td colspan="2" BGCOLOR="#D7FFD7"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;">
          <strong>Author</strong>
        </td>
       </tr>
       <tr>
        <TD COLSPAN="2" BGCOLOR="#FFFFD0"
STYLE="width:391;text-align:left;font-family:Arial;font-
size:10pt;" >
          <BLOCKQUOTE>
          <xsl:for-each select="book_description/author">
            <xsl:value-of />;
          </xsl:for-each>
          </BLOCKQUOTE>
        </TD>
      </tr>
      <xsl:apply-templates select="book_description/publisher" />
      <tr>
        <td bgColor="#D7FFD7" STYLE="font-family:Arial;
text-align:left;width:196;font-size:10pt">
          <strong>Edition</strong>
        </td>
        <td bgColor="#D7FFD7" STYLE="text-align:left;
width:195;font-family:Arial;font-size:10pt">
          <strong>Price</strong>
        </td>
      </tr>
      <tr>
        <xsl:apply-templates select="book_description/edition" />
        <xsl:apply-templates select="book_description/price" />
      </tr>
      <xsl:apply-templates select="book_description/review" />
    </TABLE>
    <p>
      <table width="400" height="80" bgcolor="#FFFFFF" >
        <tr>
          <td STYLE="text-align:center">
            <input TYPE="Image" SRC="images/buy.gif"
WIDTH="112" HEIGHT="24" BORDER="0" ALT="Add to Cart" ALIGN="Middle" />
          </td>
        </tr>
      </table>
    </p>
```

```
   </BODY>
  </HTML>
</xsl:template>
<!— Шаблоны для замещения —>

<!— Шаблон заголовка книги—>
<xsl:template match="book_title">
  <tr>
   <td colspan="2" BGCOLOR="#D7FFD7"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;">
     <strong>Book Title</strong>
   </td>
  </tr>
  <tr>
   <TD COLSPAN="2" BGCOLOR="#FFFFD0"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;" >
     <BLOCKQUOTE>
      <xsl:value-of />
     </BLOCKQUOTE>
   </TD>
  </tr>
</xsl:template>
<!— Конец шаблона —>

<!— Шаблон названия издательства —>
<xsl:template match="publisher">
  <tr>
    <td colspan="2" BGCOLOR="#D7FFD7"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;">
    <strong>Publisher</strong>
   </td>
  </tr>
  <TR>
    <TD COLSPAN="2" BGCOLOR="#FFFFD0"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;" >
    <BLOCKQUOTE>
      <xsl:value-of />
     </BLOCKQUOTE>
    </TD>
  </TR>
</xsl:template>
<!— Конец шаблона —>

<!— Шаблон даты издания —>
<xsl:template match="edition">
  <TD BGCOLOR="#FFFFD0" STYLE="width:391;text-
```

```
align:left;font-family:Arial;font-size:10pt;" >
    <BLOCKQUOTE>
     <xsl:value-of />
    </BLOCKQUOTE>
  </TD>
</xsl:template>
<!– Конец шаблона –>

<!– Шаблон стоимости –>
<xsl:template match="price">
  <TD BGCOLOR="#FFFFD0" STYLE="width:391;text-align:left;
font-family:Arial;font-size:10pt;" >
    <BLOCKQUOTE>
     $<xsl:value-of />
   </BLOCKQUOTE>
  </TD>
</xsl:template>
<!– Конец шаблона –>

<!– Шаблон аннотации–>
<xsl:template match="review">
  <tr>
    <td colspan="2" BGCOLOR="#D7FFD7"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;">
     <strong>Publisher</strong>
    </td>
  </tr>
  <TR>
    <TD COLSPAN="2" BGCOLOR="#FFFFD0"
STYLE="width:391;text-align:left;font-family:Arial;font-size:10pt;" >
     <BLOCKQUOTE>
       <xsl:value-of />
     </BLOCKQUOTE>
    </TD>
  </TR>
</xsl:template>
<!– Конец шаблона –>

</xsl:stylesheet>
```

Глава 8

purchase_order.htm

```
<HTML>
  <HEAD><TITLE>Purchase Order</TITLE></HEAD>
  <BODY Width="600" BGCOLOR=#3399FF>
```

```
<P STYLE="font-family:Arial;text-align:center;font-size:20pt;">
  <FONT COLOR=#990066><B>Purchase Order</B></FONT>
</P>
<FORM ACTION="purchase_order.asp" METHOD="Post">
<P ALIGN="CENTER">
<TABLE Border="0" Cellpadding="0" Cellspacing="0">
  <TR>
    <TD Style="font-family:Arial;font-size:10pt;
        font-weight:bold;text-align:left;width:100pt;">
      Item Code
    </TD>
    <TD>
      <INPUT TYPE=TEXT NAME="Item_code">
    </TD>
  </TR>
  <TR>
    <TD Style="font-family:Arial;font-size:10pt;
        font-weight:bold;text-align:left;">
      Item Name
    </TD>
    <TD>
      <INPUT TYPE=TEXT NAME="Item_name">
    </TD>
  </TR>
  <TR>
    <TD Style="font-family:Arial;font-size:10pt;
        font-weight:bold;text-align:left;">
      Quantity
    </TD>
    <TD>
      <INPUT TYPE=TEXT NAME="Quantity">
    </TD>
  </TR>
  <TR>
    <TD colspan="2">
       
    </TD>
  </TR>
  <TR>
    <TD Align="right">
      <INPUT TYPE="SUBMIT" VALUE="Submit Form"> 
    </TD>
    <TD Align="left">
       <INPUT TYPE="RESET" VALUE="Clear Form">
    </TD>
  </TR>
```

```
    </TABLE>
    </P>
    </FORM>
  </BODY>
</HTML>
```

purchase_order.sct

```
<scriptlet>
  <registration
Description="This scriptlet is used to write a
purchase order from form input"
    ProgID="Purchase_order.Scriptlet"
    Version="1.0"
    classid="{8D9CC880-D79F-11d2-B7C8-00C0DFE39736}"
  >
  </registration>
    <public>
      <method name="hello" />
    </public>

  <script language="VBSCript">
    public function hello(code, name, howmany)
      dim pur_code, pur_nam, pur_quant
      pur_code = code
      pur_nam = name
      pur_quant = howmany

      Set fso =CreateObject("Scripting.FileSystemObject")
      Set MyFile =fso.CreateTextFile("c:\purchase_order.txt", True)
      MyFile.WriteLine(pur_code)
      MyFile.WriteLine(pur_nam)
      MyFile.WriteLine(pur_quant)
      MyFile.Close
    end function
  </script>
</scriptlet>
```

purchase_order.asp

```
<%
set oData= createobject("Purchase_order.Scriptlet")
 rs=oData.hello(request.form("item_code"), request.form _
 ("item_name"),request.form("quantity"))
 response.write("your purchase has been noted. We will "&
 "be shipping it to you soon.")
%>
```

Глава 9

edi.asp

```
<SCRIPT Language ="VBScript" RUNAT="Server">
  set Change = CreateObject("xmliser.Scriptlet")
  Change.makeXML()
  Response.Write("Conversion to XML complete.")
</SCRIPT>
```

edi.sct

```
<scriptlet>
  <registration
    Description="This scriptlet is used to convert a 801
Transaction Set into XML"
    ProgID="xmliser.Scriptlet"
    Version="1.0"
    classid="{550A4720-54BF-11d3-A476-008048EA88E1}"
  >
  </registration>
    <public>
      <method name="makeXML" />
    </public>

  <script Language = "VBScript">
    public sub makeXML()
      Const ForReading = 1
      Const TristateUseDefault = -2, TristateTrue = -1, TristateFalse = 0
      Dim fs, f, ts, s

      Set fs = CreateObject("Scripting.FileSystemObject")
      Set f = fs.GetFile("C:\Inetpub\wwwroot\edi.txt")
      Set ts = f.OpenAsTextStream(ForReading, TristateUseDefault)
      Storage = ts.ReadAll
      ts.Close

      Dim num, i

     num = InStr(1, Storage, "N1*BT*")
     num = num - 1
     t = Left(Storage, num)
     Storage = Replace(Storage, t, " ")

     leng1 = Len(Storage)

     num = InStr(1, Storage, "SE*")
     num = leng1 - num + 1
```

```
    t = Right(Storage, num)
    Storage = Replace(Storage, t, " ")

    newl = "<organization>" & Chr(13) & "<name>"
    Storage = Replace(Storage, "N1*BT*", newl)
    newl = "</name>" & Chr(13) & "<street>"
    Storage = Replace(Storage, "~ N3*", newl)
    newl = "</street>" & Chr(13) & "<state>"
    Storage = Replace(Storage, "~ N4*", newl)
    newl = "</state>" & Chr(13) & "</organization>" & _
          Chr(13) & "<organization>"
     Storage = Replace(Storage, "~ <organization>", newl)
     newl = "</state>" & Chr(13) & "</organization>" & _
          Chr(13) & Chr(13) & "<contact>" & Chr(13) & "<name>"
     Storage = Replace(Storage, "PER*AD*", newl)
     newl = "</name>" & Chr(13) & "<phone>"
     Storage = Replace(Storage, "*TE*", newl)
     newl = "</phone>" & Chr(13) & "</contact>" & _
          Chr(13) & Chr(13) & "<order>" & Chr(13)
     Storage = Replace(Storage, "~ ITD*01*3*2**10", newl)
     newl = "<item>" & Chr(13) & "<quantity>"
     Storage = Replace(Storage, "~ IT1** ", newl)
     newl = "</quantity>" & Chr(13) & "</item>" & _
          Chr(13) & "<total_cost>"
     Storage = Replace(Storage, "~ TDS* ", newl)
     newl = Chr(13) & "</total_cost>" & Chr(13) & "</order>" & _
          Chr(13) & Chr(13) & "<shipper>" & Chr(13) & "<name>"
     Storage = Replace(Storage, "~ CAD*M**** ", newl)
     newl = "</name>" & Chr(13) & "</shipper>" & Chr(13) & "</invoice>"
     Storage = Replace(Storage, "~ CTT* 4* 20~", newl)
     newl = "0" & "</quantity>" & Chr(13) & "</item>" & _
          Chr(13) & Chr(13) & "<item>"
     Storage = Replace(Storage, "0<item>", newl)
     newl = "</quantity>" & Chr(13) & "<code>"
     Storage = Replace(Storage, "*VC* ", newl)
     Storage = Replace(Storage, "*", ",")
     Storage = Replace(Storage, "~", "")

     Set fs = CreateObject("Scripting.FileSystemObject")
     Set MyFile = fs.CreateTextFile("c:\edi-new.xml", True)
     MyFile.WriteLine ("<?xml version='1.0' ?>")
     MyFile.WriteLine ("<invoice>")
     MyFile.Write (Storage)
     MyFile.Close
   End Sub
  </script>
</scriptlet>
```

edi.txt

```
ISA**00*0000000000*01*01*PASSWORDME*01*123456789 987654321
890714*2210*U*000000008*O*P*~
GS*IN*012345678*087654321*900509*2210*000001*X*002040~
ST*801*0001~74832 BEG*940606*1045*940606*~ N1*BT* RAMONA
PUBLISHING~ N3* P.O. BOX 679342~
N4* TX* 77234~ N1*BT* HANSEL PUBLISHING~ N3* 101 APPLE PIE
ST.~ N4* NY* 00103~ N1*BT*
FIVELAKES PUBLISHING~ N3* 79 RIVER DRIVE~ N4* TX* 74564~
PER*AD* J. DOE*TE* 2104355445~
ITD*01*3*2**10~ IT1** 3* CA* 12.75**VC** 6900~ IT1** 12*
EA* 2.99**VC** P450~ IT1** 4*
EA* 5.99**VC** 1640~ IT1** 1* DZ* 2.45**VC** 1507~ TDS*
100.54~ CAD*M**** CONSOLIDATED TRUCK~
CTT* 4* 20~ SE*21*000001~ GE*1*000001~ IEA*1*000000008~
```

edi.xml

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="edi.xsl" ?>
<invoice number="1045">
<organization status="send to">
<name>RAMONA PUBLISHING</name>
<street>P.O. BOX 679342</street>
<city>HOUSTON</city>
<state>TX</state>
<zip>77234</zip>
</organization>
<organization status="charge">
<name>HANSEL PUBLISHING</name>
<street>101 APPLE PIE ST.</street>
<city>NEW YORK</city>
<state>NY</state>
<zip>00103</zip>
</organization>
<organization status="sent from">
<name>FIVELAKES PUBLISHING</name>
<street>79 RIVER DRIVE</street>
<city>DALLAS</city>
<state>TX</state>
<zip>74564</zip>
</organization>
<shipper>
<name>CONSOLIDATED TRUCK</name>
<street>534 FLAT MEADOWS</street>
```

```
<city>HOUSTON</city>
<state>TX</state>
<zip>77544</zip>
</shipper>

<contact>
<name>J. DOE</name>
<phone>2104355445</phone>
</contact>
<order>
<item>
  <code>6900</code>
  <quantity>3</quantity>
  <unit>CA</unit>
  <price_per_unit>12.75</price_per_unit>
</item>
<item>
  <code>P450</code>
  <quantity>12</quantity>
  <unit>EA</unit>
  <price_per_unit>2.99</price_per_unit>
</item>
<item>
  <code>1640</code>
  <quantity>4</quantity>
  <unit>EA</unit>
  <price_per_unit>5.99</price_per_unit>
</item>
<item>
  <code>1507</code>
  <quantity>1</quantity>
  <unit>DZ</unit>
  <price_per_unit>2.45</price_per_unit>
</item>
<total_cost>100.54</total_cost>
<date>
  <day>20</day>
  <month>4</month>
</date>
</order>
</invoice>
```

edi.xsl

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
ISA**00*0000000000*01*01*PASSWORDME*01*123456789 987654321
```

```
890714*2210*U*000000008*O*P*~
GS*IN*012345678*087654321*900509*2210*000001*X*002040~
ST*801*0001~74832
BEG*940606*1045*940606*~
<xsl:apply-templates select="invoice/organization" />
<xsl:apply-templates select="invoice/contact" />
ITD*01*3*2**10~
<xsl:apply-templates select="invoice/order/item" />TDS*
<xsl:value-of select="invoice/order/total_cost" />~
<xsl:apply-templates select="invoice/shipper" />
<xsl:apply-templates select="invoice/order/date" />
SE*21*000001~
GE*1*000001~
IEA*1*000000008~
</xsl:template>
<xsl:template name="who" match="organization">
N1*BT*
<xsl:value-of select="name" />~ N3*
<xsl:value-of select="street" />~ N4*
<xsl:value-of select="state" />*
<xsl:value-of select="zip" />~
</xsl:template>

<xsl:template match="contact">
PER*AD*
<xsl:value-of select="name" />*TE*
<xsl:value-of select="phone" />~
</xsl:template>

<xsl:template match="item">
IT1**
<xsl:value-of select="quantity" />*
<xsl:value-of select="unit" />*
<xsl:value-of select="price_per_unit" />**VC**
<xsl:value-of select="code" />~
</xsl:template>

<xsl:template match="shipper">
CAD*M****
<xsl:value-of select="name" />~
</xsl:template>
<xsl:template match="date">
CTT*
<xsl:value-of select="month" />*
<xsl:value-of select="day" />~
</xsl:template>

</xsl:stylesheet>
```

edi-new.xml

```
<?xml version='1.0' ?>
<invoice>
 <organization>
<name> RAMONA PUBLISHING</name>
<street> P.O. BOX 679342</street>
<state> TX, 77234</state>
</organization>
<organization>
<name> HANSEL PUBLISHING</name>
<street> 101 APPLE PIE ST.</street>
<state> NY, 00103</state>
</organization>
<organization>
<name> FIVELAKES PUBLISHING</name>
<street> 79 RIVER DRIVE</street>
<state> TX, 74564 </state>
</organization>
<contact>
<name> J. DOE</name>
<phone> 2104355445</phone>
</contact>

<order>
<item>
<quantity>3, CA, 12.75,,VC,, 6900</quantity>
</item>

<item>
<quantity>12, EA, 2.99,,VC,, P450</quantity>
</item>

<item>
<quantity>4, EA, 5.99,,VC,, 1640</quantity>
</item>
<item>
<quantity>1, DZ, 2.45,,VC,, 1507</quantity>
</item>
<total_cost>100.54
</total_cost>
</order>

<shipper>
<name>CONSOLIDATED TRUCK</name>
</shipper>
</invoice>
```

ПРИЛОЖЕНИЕ B

Руководство по установке

Внимание!

Программы этого курса были разработаны и протестированы с помощью Microsoft Site Server 3.0 Commerce Edition и Microsoft SQL Server 7.0 в системе Microsoft Windows NT Server 4.0. Для работы Site Server Commerce в Windows 2000 требуется другое программное обеспечение. Для получения последней информации посетите Web-страницу **http://www.microsoft.com/siteserver/commerce**.

Требования к компьютеру

Для выполнения всех работ данного курса необходим один компьютер, за исключением работ, содержащихся в папке DemoCode\Mod05, носящих необязательный характер. Для этих работ требуется дополнительный компьютер. Прежде чем приступить к работе, подготовьте компьютер в соответствии с данной инструкцией.

Аппаратное обеспечение

Компьютер должен отвечать следующим требованиям:

- процессор — Pentium II 300 МГц;
- оперативная память (RAM) — 128 Мбайт;
- жесткий диск — 4 Гбайт;
- 12X CD-ROM;
- сетевая карта (Network adapter);
- видеоадаптер 4 Мбайт;
- монитор Super VGA (SVGA), 17 дюймов;

- ❒ мышь Microsoft Mouse (или совместимое с ней устройство) и клавиатура;
- ❒ звуковая карта;
- ❒ доступ к Internet.

Программное обеспечение

Для инсталляции на компьютере требуется следующее программное обеспечение. (При запуске программы `setup` следуйте приведенной ниже последовательности. Прочитайте раздел данного руководства для каждого указанного компонента.)

- ❒ Microsoft Windows NT 4.0 Server
- ❒ Microsoft Visual Studio 6.0 Enterprise Edition
  - Windows NT Server 4.0 Service Pack 3
  - Microsoft Internet Explorer 4.01 SP1
  - Microsoft Visual Basic 6.0
  - Microsoft Visual InterDev 6.0 (не обязательно)
  - MSDN Library (не обязательно)
- ❒ Windows NT 4.0 Option Pack
- ❒ Microsoft FrontPage 98 Server Extensions v. 3.0.2.1706
- ❒ Windows NT Server 4.0 Service Pack 4
- ❒ Internet Explorer 5.0
- ❒ Microsoft SQL Server 7.0 Standard Edition
- ❒ Microsoft Data Access 2.1 (MDAC)
- ❒ Microsoft Site Server 3.0
- ❒ Microsoft Site Server 3.0 Commerce Edition
- ❒ Site Server 3.0 Service Pack 2
- ❒ Active Directory Services Interface 2.5 (ADSI)
- ❒ Commerce Interchange Pipeline Manager
- ❒ Windows NT Server 4.0 Service Pack 5
- ❒ Файлы курса лабораторных работ
- ❒ Файлы баз данных FiveLakes и Ramona

Конфигурация компьютера

Настройте компьютер как отдельный сервер (stand-alone server), входящий в рабочую группу WORKGROUP.

Примечание

Если ваш компьютер будет участвовать в подключении к локальной сети, и вы назначили статические IP-адреса вместо использования Dynamic Host Configuration Protocol (DHCP), убедитесь, что имя и адрес, который вы назначили компьютеру, является уникальным. Это позволит избежать конфликтов в сети.

Инструкция по установке

Подготовка к инсталляции программного обеспечения

Вам потребуется сделать следующее:

- прочитать о последовательности выполнения установки каждого компонента;
- настроить аппаратное обеспечение в соответствии с инструкцией изготовителя;
- отформатировать диск C: с разделом не менее чем 4 Гбайт;
- собрать все необходимое программное обеспечение.

Внимание!

Устанавливайте все программное обеспечение на диск C:. В противном случае, вам может потребоваться изменить ряд инструкций к лабораторным работам (указать соответствующий путь).

Примечание

Время, необходимое для выполнения установки на одном компьютере, — 6 часов.

Windows NT 4.0 Server

Требования по установке:

- установите тип сервера — stand-alone server (выделенный сервер);
- отформатируйте разделы под файловую систему Windows NT File System (NTFS);
- инсталлируйте Windows NT 4.0 на диск C:;
- не устанавливайте Internet Information Server 2.0 (IIS).

Для того чтобы инсталлировать Windows NT 4 Server, необходимо выполнить действия, перечисленные в табл. B1.

Таблица В1. *Действия по инсталляции Windows NT 4 Server*

Параметры, указываемые программой установки	**Выбор значения параметра**
Partition choice (выбор раздела на диске)	Выберите диск C:
Format options (опции форматирования)	Преобразуйте область для установки в NFTS
Install directory choice (выбор каталога для инсталляции)	Используйте каталог, заданный по умолчанию
NT will examine disks for corruption (NT будет осуществлять проверку дисков на наличие повреждений)	Необязательно
Name and organization (имя и организация)	Введите данные в соответствующие поля
CD Key field (поле ключа компакт-диска)	Введите ключ компакт-диска
Диалоговое окно для указания **licensing modes** — количества лицензий (предлагаемое количество зависит от инсталляционных компакт-дисков)	Рекомендуется 100 параллельных подключений. Руководствуйтесь своим лицензионным соглашением
Computer Name (имя компьютера)	Введите любое допустимое имя
Server Type (тип сервера)	Stand-Alone Server (выделенный сервер)
Password for the Administrator account (пароль для учетной записи администратора)	Оставьте поле для пароля пустым
Emergency Repair Disk (предложение создать аварийный диск)	Необязательно
Select Components dialog box (диалоговое окно выбора компонентов	Используйте значения, принятые по умолчанию
Is now ready to install Windows NT Networking (программа Windows NT Setup готова приступить к установке сетевых средств Windows NT)	Используйте значения, принятые по умолчанию
This computer will participate on a network (данный компьютер будет работать в сети) и **Wired to the network** (подключен к сети)	Используйте значения, соответствующие вашему оборудованию
Internet Information Server 2.0	Не устанавливайте IIS 2.0. Это сэкономит ваше время, т. к. IIS 4.0 устанавливается вместе с NT Option Pack 4

Таблица B1 (окончание)

Параметры, указываемые программой установки	Выбор значения параметра
Network Adapter (сетевая карта)	Используйте значения, подходящие для вашего компьютера
Network Protocols (сетевые протоколы)	Используйте TCP/IP
Network Services (сетевые службы)	Используйте значения, заданные по умолчанию
TCP/IP Setup (установка TCP/IP)	По возможности, используйте DHCP
Network Bindings (сетевая привязка)	Используйте значения, заданные по умолчанию
Start the network (запуск сети)	Используйте значения, заданные по умолчанию
Domain or Workgroup (домен или рабочая группа)	Используйте рабочую группу, заданную по умолчанию
Date/Time Properties (установка даты/времени)	Установите текущую дату и время, а также вашу временную зону (часовой пояс)

Конфигурирование драйверов

Для того чтобы установить драйверы аппаратного обеспечения, подключите аудио- и видеодрайверы в соответствии с инструкцией изготовителя. Подключите драйвер для карты сетевого адаптера (NIC, Network Interface Card), если это не было сделано на предыдущих этапах. Обратите внимание, что для обеспечения корректной работы некоторого аппаратного обеспечения требуется, чтобы первоначально, перед установкой драйверов была осуществлена инсталляция Windows NT Service Pack 3.

Важно, чтобы прежде чем приступить к инсталляции Site Server Commerce Edition, вы установили соответствующий видеодрайвер, а также значение разрешения экрана — 800×600 пикселов.

Для установки режима отображения:

1. В меню кнопки **Start** выберите пункт **Settings**, а затем — команду **Control Panel**. В появившемся окне дважды щелкните по значку **Display**.
2. В диалоговом окне **Display Properties** перейдите на вкладку **Settings**.
3. В раскрывающемся списке **Colors** выберите значение **256 Colors**, установите ползунок **Screen area** в положение **800 by 600 pixels**, а затем нажмите кнопку **OK**.

Инсталляция программного обеспечения из состава Visual Studio 6.0 Enterprise Edition

Используйте Visual Studio 6.0 для выполнения эффективной установки Windows NT Service Pack 3, Internet Explorer 4.01, Visual Basic 6.0 и MSDN Library.

Для инсталляции Microsoft Visual Studio 6.0:

1. Запустите исполняемый файл setup для Visual Studio.
2. Введите личные и лицензионные данные в соответствующие диалоговые окна.
3. Если появится предложение по обновлению программного обеспечения, то согласитесь с установками, принятыми по умолчанию. Для каждого программного продукта введите лицензионную и другую пользовательскую информацию:
   - обновите Microsoft Virtual Machine for Java. Мастер инсталляции предложит выполнить перезагрузку компьютера;
   - установите Windows NT 4.0 Service Pack 3, указав параметры, принятые по умолчанию. Создание каталога для деинсталляции необязательно;
   - установите Internet Explorer 4.01, используя параметры по умолчанию.
4. Инсталлируйте Visual Studio 6.0 в каталог, заданный по умолчанию. Выполните выборочную (custom) установку компонентов.
5. В окне **Visual Studio 6.0 Enterprise — Custom**:
   - перечисленные ниже программы не требуются. Для того чтобы они не устанавливались, сбросьте соответствующие флажки:
     - Microsoft Visual C++;
     - Microsoft Visual FoxPro 6.0;
     - Microsoft Visual SoureSafe 6.0;
     - ActiveX.
   - установите флажок для инсталляции Microsoft Visual Basic 6.0;
   - установка Microsoft Visual InterDev 6.0 является необязательной.

Примечание

Установка MSDN Library является необязательной, но мы настоятельно рекомендуем выполнить ее.

6. Если вы решите инсталлировать MSDN, примите значения, заданные по умолчанию, прежде чем перейдете в окно **Installation options**, а затем

щелкните кнопку **Custom**. Проверьте следующие параметры, а потом щелкните кнопку **Continue**:

- Full Text Search Index (полнотекстовый индекс для поиска);
- VB Documentation (документация по VB);
- VID Documentation (документация по VID, если вы установили Visual InterDev);
- VS Shared Documentation (документация по VS Shared).

7. Выполните инсталляцию Visual Studio, используя параметры, заданные по умолчанию. При этом:
   - инсталляция InstallShield не требуется;
   - не запускайте мастер инсталляции BackOffice (BackOffice Installation Wizard);
   - зарегистрируйте Visual Studio в соответствии с вашими конкретными условиями.

Windows NT 4.0 Option Pack

Обратите внимание на следующие моменты в инсталляции:

- ❒ выполняйте выборочную (custom) инсталляцию;
- ❒ не устанавливайте FrontPage Server Extensions.

Примечание

Если во время инсталляции NT Server 4.0 вы установили IIS 2.0, то необходимые шаги будут отличаться от тех, которые здесь приведены. Если IIS 2.0 уже был установлен, то воспользуйтесь параметром **Upgrade Plus installation**, а затем выберите **Custom installation**. Среди опций, предложенных в окне **Custom installation**, вы должны отметить нужную для инсталляции Index Server.

Для того чтобы произвести инсталляцию Windows NT 4.0 Option Pack:

1. Запустите исполняемый setup-файл для Option Pack. Последовательно выполните действия, указанные в окнах приглашения и лицензионного соглашения.
2. В диалоговом окне **Installation type** нажмите кнопку **Custom**.
3. В диалоговом окне **Select Component** сбросьте флажок **FrontPage 98 Server Extensions**. Проверьте, чтобы был выбран **Index Server**, и затем щелкните кнопку **Next**.
4. Продолжайте инсталляцию, принимая в остальных диалоговых окнах значения, заданные по умолчанию.

FrontPage Server Extensions

Для инсталляции FrontPage Server Extensions:

1. Запустите программу установки FrontPage Server Extensions.
2. Для выполнения инсталляции используйте опции, заданные по умолчанию. Введите лицензионную информацию.

Примечание

Используйте версию 3.0.2.1706 (fp98ext_x86_enu.exe), которую можно переписать по адресу: **http://officeupdate.microsoft.com/frontpage/wpp/license.htm**. Серверные расширения FrontPage 2000 могут не подходить для данной версии.

Windows NT Server 4.0 Service Pack 4

Для инсталляции Windows NT Server 4.0 Service Pack 4:

1. Запустите программу установки Windows NT Server 4.0 Service Pack 4.
2. Выполните действия, указанные в окнах приветствия и лицензионного соглашения.
3. Решение о создании резервных копий файлов является необязательным. Для сохранения дискового пространства рекомендуется отказаться от этой опции.
4. По завершении инсталляции вновь появится диалоговое окно **Windows NT Service Pack 4 Setup**. Щелкните кнопку **Restart**.
5. После перезапуска компьютера не устанавливайте обновления для 2000 года, когда для этого поступит соответствующее предложение. Обновления Y2K[1] будут установлены в одном из последующих шагов.

Internet Explorer 5.0

Обратите внимание на следующие моменты:

- ❒ выберите минимальную инсталляцию;
- ❒ включите в состав инсталлируемых компонентов Microsoft Virtual Machine;
- ❒ включите Microsoft Wallet.

Для инсталляции Internet Explorer 5.0:

1. Запустите программу установки Internet Explorer.
2. Появится приветственное окно **Welcome to Setup for Internet Explorer and Internet Tools**. Если вы согласны с условиями лицензионного соглашения, щелкните кнопку **Next**.

---

[1] Проблема Y2K возникла в связи со сменой дат, т. е. переходом к 2000 г. — *Ред.*

3. В диалоговом окне **Windows update** выберите опцию **Install Minimal, or Customize your browse**, а затем нажмите кнопку **Next**.
4. В диалоговом окне **Component Options** просмотрите весь список компонентов:
   - под заголовком Internet Explorer 5 выберите компонент **Microsoft virtual machine**;
   - под заголовком Additional Components выберите **Wallet**;
   - щелкните по кнопке **Next**.
5. В диалоговом окне **Restart computer** нажмите кнопку **Finish**.

SQL Server 7.0 Standard Edition

Обратите внимание на следующее:

- ❒ используйте Standard Edition;
- ❒ выполняйте локальную инсталляцию;
- ❒ в диалоговом окне **Services Accounts** оставьте поле для пароля пустым;
- ❒ после инсталляции SQL запустите службу MSDTC и установите режим его автоматического запуска.

Для установки SQL Server 7.0 Standard Edition:

1. Запустите программу установки SQL Server 7.0.
2. В открывшемся диалоговом окне выберите опцию **Install SQL Server 7.0 Components**.
3. В диалоговом окне **Install SQL Server 7.0 Components** укажите опцию инсталляции **Database Server — Standard Edition**.
4. В диалоговом окне **Select Install Method** примите опцию **Local Install**, заданную по умолчанию, а затем нажмите кнопку **Next**.
5. В диалоговом окне **Welcome** нажмите кнопку **Next**.
6. В диалоговом окне **Software License Agreement** щелкните по кнопке **Yes**, если вы принимаете условия лицензионного соглашения.
7. В диалоговом окне **User Information** проверьте правильность информации, а затем щелкните по кнопке **Next**.
8. В диалоговом окне **Setup** введите ключ компакт-диска, а затем щелкните кнопку **OK**. В окне сообщения **Setup product ID confirmation** снова нажмите кнопку **OK**.
9. В диалоговом окне **Setup Type** выберите переключатель **Typical**. Примите каталоги для инсталляции, заданные по умолчанию, а затем нажмите кнопку **Next**.

10. В диалоговом окне **Services Accounts** примите установки, заданные по умолчанию, оставьте поле для пароля пустым, а затем нажмите кнопку **Next**.
11. В диалоговом окне **Start Copying Files** нажмите кнопку **Next**.
12. В диалоговом окне **Licensing** выберите опцию **I agree that**, а затем нажмите кнопку **Continue**.
13. В некоторых вариантах инсталляции может последовать вопрос о количестве пользователей. В этом случае укажите 25 пользователей.

Примечание

В зависимости от того, какой компакт-диск вы используете для инсталляции — MSDN или же розничную версию SQL 7.0 — на этом этапе вы можете увидеть диалоговое окно **Choose licensing mode**. Если оно появится, щелкните **Add licenses**, а затем измените их количество до 25.

14. После завершения установки файлов нажмите кнопку **Finish**.
15. Выйдите из программы установки SQL Server 7.0.

Для установки автоматического запуска службы MSDTC:

1. В меню кнопки **Start** укажите пункт **Settings**, а затем — команду **Control Panel**. Дважды щелкните по значку **Services**.
2. В панели управления **Services** просмотрите список служб и выберите **MSDTC**.
3. Нажмите кнопку **Startup**.
4. Установите значение режима запуска (Startup Type) — **Automatic**.
5. Щелкните по кнопке **OK**.
6. Нажмите кнопку **Start**.
7. Произойдет запуск службы. Если он пройдет успешно, закройте окно **Services**.

Microsoft Data Access 2.1

Для инсталляции MDAC 2.1:

1. Запустите программу установки Microsoft Data Access 2.1. Выполните полную инсталляцию.
2. Когда поступит соответствующее приглашение, перезапустите Windows.

Совет

MDAC 2.1.1.3711.11 (GA) можно переписать по адресу **http://www.microsoft.com /data/download2.htm**.

Создание и конфигурирование баз данных SQL Server

Для того чтобы создать базы данных SQL для сайтов FiveLakes и Ramona:

1. Скопируйте файлы базы данных FiveLakes и Ramona из папки SampleSite, находящейся на компакт-диске, в папку C:\MSSQL7\Backup.
2. В меню кнопки **Start** укажите пункт **Programs**, а затем — команду **Microsoft SQL Server 7.0**, и далее команду **Enterprise Manager**.
3. В левой панели дважды щелкните мышью на элементе **Microsoft SQL Servers**.
4. Дважды щелкните мышью на **SQL Server Group**.
5. Откройте группу, находящуюся в списке **SQL Server Group**, соответствующую имени вашего компьютера.

Примечание

Если появится диалоговое окно **Internet Explorer Install on Demand**, для каждого из этих компонентов выберите режим **Never download**, а затем щелкните по кнопке **OK**.

6. Раскройте дерево папки **Databases**, а затем выберите элемент **Databases**.
7. Выполните следующие шаги для будущего сайта FiveLakes, а затем повторите их для сайта Ramona.
   - Щелкните правой кнопкой мыши в правой панели. Появится список. Выберите команду **New Database**.
   - В поле **Name** введите имя базы данных, которую вы устанавливаете в настоящий момент — `FiveLakes` или `Ramona`.
   - В группе **File growth** установите переключатель **In megabytes**, а в соседнем поле введите `10`.
   - Нажмите кнопку **OK**.

Для того чтобы восстановить базы данных FiveLakes и Ramona:

1. Щелкните правой кнопкой мыши по файлу базы данных. Выберите команду **All Tasks**, а затем команду **Restore Database**.
2. В диалоговом окне **Restore Database** перейдите на вкладку **Options**.
3. Установите флажок **Force restore over existing database**.
4. Перейдите на вкладку **General**.
5. Выберите переключатель **Restore backup sets from device(s)**.
6. Нажмите кнопку **Select Devices**.
7. Щелкните по кнопке **Add**.

8. Щелкните кнопку пути с изображением многоточия. В папке C:\MSSQL7 \Backup выберите соответствующую базу данных и затем нажмите кнопку **OK**.
9. Щелкните **OK**, чтобы принять значения, заданные по умолчанию в диалоговом окне **Choose Restore Destination**.
10. В диалоговом окне **Assign Restore Devices** снова щелкните кнопку **OK**.
11. Нажмите кнопку **OK** для восстановления базы данных. Появится индикатор выполнения процесса. Когда база данных будет восстановлена, вновь нажмите кнопку **OK**.
12. Если на этом шаге появится сообщение об ошибке, то вы, возможно, пропустили шаг 3 или не установили флажок **Force restore over existing database**.
13. После восстановления баз данных `FiveLakes` и `Ramona` закройте SQL Server Enterprise Manager.

Примечание

Вы должны выполнить указанные действия дважды, чтобы восстановить базу данных `FiveLakes` и базу данных `Ramona`.

Примечание

После указания имени базы данных, места ее данных, в том числе и регистрационных — все эти параметры нельзя будет изменить. Если вы допустили опечатку, а затем приняли изменения, вам следует удалить базу данных и создать ее снова.

Создание System DSN для каждой базы данных (запустите данную процедуру сначала для FiveLakes, а затем для Ramona):

1. В меню кнопки **Start** выберите пункт **Settings**, затем команду **Control Panel**. В появившемся окне дважды щелкните по значку **ODBC Data Sources**.
2. В диалоговом окне **ODBC Data Source Administrator** перейдите к вкладке **System DSN**, а затем нажмите кнопку **Add**.
3. В диалоговом окне **Create New Data Source** в списке **Select a driver** выберите **SQL Server**, а потом нажмите кнопку **Finish**.
4. В диалоговом окне **Create a New Data Source to SQL Server** введите точное имя базы данных (`FiveLakes` или `Ramona`) в поле **Name**. В раскрывающемся списке **Server** выберите значение **local**, а затем щелкните по кнопке **Next**.
5. Выберите опцию **With SQL Server authentication using a login ID and password entered by the user**.

6. В поле **Login ID** введите `sa`, оставьте поле для пароля пустым, а затем нажмите кнопку **Next**.
7. Отметьте флажок **Change the default database to**, потом выберите соответствующую базу данных (`FiveLakes` или `Ramona`) из раскрывающегося списка, а затем щелкните по кнопке **Next**.
8. Нажмите кнопку **Finish**.
9. В диалоговом окне **ODBC Microsoft SQL Server Setup** укажите **Test Data Source** для проверки возможности подключения к данной базе данных. Если тесты выполнены успешно, щелкните по кнопке **OK**. В противном случае, просмотрите информацию об ошибках, чтобы найти вероятные причины.
10. В диалоговом окне **ODBC Microsoft SQL Server Setup** нажмите кнопку **OK**.
11. В диалоговом окне **Data Source Administration** щелкните по кнопке **OK**, чтобы завершить создание DSN.

Site Server 3.0

Обратите внимание на следующее:

- выберите **Custom installation**, чтобы включить поддержку базы данных SQL Server;
- создайте учетные записи для опций **Publishing** и **Search**.

Для инсталляции Site Server 3.0:

1. Запустите программу установки Site Server 3.0.
2. В открывшемся окне укажите **Server Installation**.
3. Введите соответствующую информацию в поля лицензирования, регистрации и информации о пользователе. Для инсталляции используйте путь, заданный для умолчания.
4. В диалоговом окне **Choose Installation Type** выберите **Custom**.
5. В диалоговом окне **Select Features**:
   - щелкните мышью по знаку плюс (+), расположенном следом за надписью **Analysis**, для отображения компонентов;
   - выберите опцию **SQL Server Database Support**;
   - щелкните по кнопке **Next**.

Примечание

Если вам не удалось добавить поддержку баз данных SQL Server, продолжите инсталляцию. Когда установка Site Server 3.0 будет выполнена, перезапустите

компьютер и затем снова запустите установочную программу. Выберите режим добавления компонентов и затем укажите поддержку базы данных SQL Server (SQL Server Database Support).

6. В диалоговом окне **Specify a Program Folder** нажмите кнопку **Next**.
7. В диалоговом окне **Configure User Accounts** щелкните **Set User Account**. Оставьте оба поля для пароля пустыми, нажмите кнопку **OK**, а затем — кнопку **Next**.
8. Продолжайте инсталляцию, используя значения, заданные по умолчанию.

Примечание

При инсталляции Commerce Server 3.0 вы можете увидеть диалоговое окно, которое содержит подобное сообщение:

```
Confirm File ReplaceSource: C:\Microsoft Site
Server\_install\_WSH\vbscript
Target: C:\WINNT\System32\vbscript.dll
The target file exists and is newer than the source. Overwrite
existing file?
```

В этом диалоговом окне щелкните по кнопке **No to All**.

Site Server 3.0 Commerce Edition

Обратите внимание на следующее:

- выполните выборочную инсталляцию;
- установите Commerce Server SDK;
- не устанавливайте Ad Manager или Ad Server;
- не устанавливайте ни один из следующих сайтов с примерами: Clocktower, Microsoft Market, Microsoft Press и Volcano Coffee.

Для установки Site Server 3.0, Commerce Edition:

1. Запустите программу установки Site Server 3.0 Commerce Edition.
2. В открывшемся окне укажите **Server Installation**.
3. Введите соответствующие данные в поля лицензирования, регистрации и информации о пользователе. Используйте путь для инсталляции, заданный по умолчанию.
4. В очередном диалоговом окне щелкните **Custom**.
5. Сбросьте все флажки, относящиеся к сайтам с примерами, чтобы не допустить их инсталляции. Установите только флажок инсталляции SDK, а затем нажмите кнопку **Next**.

6. В окне **Question** щелкните по кнопке **Yes**.
7. В следующем диалоговом окне просмотрите перечень устанавливаемых дополнений. Убедитесь, чтобы в списке инсталлируемых компонентов присутствовал только SDK. Щелкните по кнопке **Next**.
8. Продолжите инсталляцию, используя значения, заданные по умолчанию.

Site Server 3.0 Service Pack 2

Для инсталляции Site Server 3.0 Service Pack 2:

1. Запустите программу установки Site Server 3.0 Service Pack 2.
2. Продолжите инсталляцию с параметрами, заданными по умолчанию.
3. Право решения о создании каталога для деинсталляции остается за вами. Для сохранения дискового пространства рекомендуется от этого отказаться.

Active Directory Services Interface 2.5

Для установки ADSI 2.5:

1. Запустите программу установки.
2. Появится диалоговое окно **ADSI Welcome**. Щелкните кнопку **Yes**.
3. В диалоговом окне **ADSI 2.5 License Agreement** щелкните **Yes,** если вы принимаете условия лицензионного соглашения.
4. В окне **ADSI installation successful** нажмите кнопку **OK**.

Совет

ADSI 2.5 для платформ x86 Intel (файл ads.exe, объемом 651 Кбайт) можно получить по адресу **http://www.microsoft.com/NTServer/nts/downloads/previews/ADSI25/default.asp**.

Commerce Interchange Pipeline Manager

Для того чтобы установить Commerce Interchange Pipeline Manager:

1. Запустите программу установки CIPM.
2. Продолжайте инсталляцию, если вы принимаете условия лицензионного соглашения.
3. Запустите опцию **Complete Install**.

Совет

Commerce Interchange Pipeline Manager (файл cipm.exe, объемом 3,9 Мбайт) можно переписать по адресу **http://www.microsoft.com/siteserver/commerce/DeployAdmin/Pipeline.htm**.

Windows NT Server 4.0 Service Pack 5

Для инсталляции Windows NT Server 4.0 Service Pack 5:

1. Запустите программу инсталляции Windows NT Server 4.0 Service Pack 5.
2. В диалоговом окне **Welcome** выберите опцию **Accept the License Agreement,** если вы принимаете условия лицензионного соглашения. Затем щелкните мышью для того, чтобы сбросить флажок **Backup files** и нажмите кнопку **Install**.
3. Когда инсталляция будет выполнена, вновь появится диалоговое окно **Windows NT Service Pack 5 Setup**. Щелкните по кнопке **Restart**.

Подготовка к созданию сайтов с примерами

Вам нужно дважды выполнить эту процедуру — для FiveLakes и для Ramona.

Для запуска Create New Site Foundation Wizard (Мастера создания новой "основы" сайта):

1. Нажмите кнопку **Start**, выберите последовательно **Programs**, **Microsoft Site Server**, **Administration**, а затем — **Site Server Service Admin (HTML)**.
2. На Web-странице **Web Based Administration** щелкните **Commerce**.
3. На Web-странице **Getting Started with Site Server Commerce**, в левом окне, щелкните **Server Administration**.
4. На Web-странице **Server Administration for localhost** щелкните **Create**.
5. Выберите **Default Web Site**, а затем щелкните кнопку **Next**.
6. Введите сокращенные и полные (те, что будут потом отображаться) имена создаваемых сайтов, используя значения в табл. В2. Убедитесь, что между словами отсутствуют пробелы. По окончании нажмите кнопку **Next**.

Таблица В2. *Полные и сокращенные имена сайтов*

Сокращенное имя	Полное (отображаемое) имя
FiveLakes	FiveLakes Publishing
Ramona	Ramona

7. В окне **Select a Directory Location** (Выбор каталога) примите значения, заданные по умолчанию. Именно здесь, в этом каталоге, Site Server создаст Web-страницы. В нем же будут установлены файлы с примерами, скопированными с компакт-диска. Щелкните по кнопке **Next**.
8. В окне **Formulate a Database Connection String**:

- в списке допустимых DSN выберите **FiveLakes** или **Ramona**. Вам возможно понадобится прокрутить этот список до конца, чтобы найти правильное DSN;
- в поле **Database Login** введите `sa`, оставьте поле для пароля пустым, а затем щелкните по кнопке **Next**.

9. В окне **Specify Manager Account** выберите опцию, используемую по умолчанию, — **Use an existing Windows NT account**, а затем нажмите кнопку **Next**.
10. В окне **Select Windows NT Domain** домен, относящийся к имени данного компьютера, выбран по умолчанию (например, \\ECOMSERVER). Оставьте эти установки и щелкните кнопку **Next**.
11. Убедитесь, что в окне **Select a Windows NT account** выбрано значение **Administrator**, а затем нажмите кнопку **Next**.
12. В окне **Finish** щелкните по кнопке **Finish**.
13. Появится сообщение **Site Creation Complete** (Создание сайта выполнено). Нажмите кнопку **OK**.

Создаем Web-сайты FiveLakes и Ramona

Вам следует запустить Site Builder Wizard (Мастер компоновки сайта) дважды — один раз для создания Web-сайта FiveLakes и второй — для создания Web-сайта Ramona.

Для запуска Site Builder Wizard:

1. На странице **Create New Site Foundation Wizard — Site Creation Complete**, щелкните по ссылке **http://localhost:80/FiveLakes/manager/default.asp** или **http://localhost:80/Ramona/manager/default.asp**.

Примечание

Если Site Builder Wizard не запускается, в поле **Address** браузера Internet Explorer введите **http://localhost:80/FiveLakes/manager/default.asp** или **http://localhost:80/Ramona/manager/default.asp** и затем нажмите клавишу <Enter>.

1. На странице **Site Manager** для нового сайта щелкните по кнопке **Site Builder Wizard**.
2. На страницах **Welcome**, **Site Type**, **Merchant Information** нажмите кнопку **Next**.
3. На странице **Locale** щелкните по кнопке **Next**, чтобы принять значение, предложенное по умолчанию — **English (United States)**.
4. На странице **Site Style** щелкните по кнопке **Next**.
5. На странице **Promotions** выберите **Price promotions** и **Cross-sell promotions**, а затем нажмите кнопку **Next**.

6. На страницах **Features**, **Product Attribute Type**, **Product Structure, Shipping & Handling**, **Tax: USA** щелкните по кнопке **Next**.
7. На странице **Payment Methods** щелкните кнопку **Next**, чтобы принять кредитные карты, заданные по умолчанию.
8. На странице **Order History** выберите **Retain order history and receipt information**, а затем нажмите кнопку **Next**.
9. На странице **Output Options** нажмите кнопку **Finish** для создания сайта.

Проверка

Для проверки, что сайт был создан успешно, когда слово **Done** появится в конце списка **Shopper Site Pages**, щелкните **Here is your shopping site**. Появится новый Web-сайт.

Права доступа для работы с каталогом FiveLakes

Назначьте каталогу FiveLakes общий уровень безопасности (sharing security level) для категории пользователей **Everyone — Full Contro**l (Полный контроль).

1. Откройте папку C:\Inetpub\wwwroot с помощью Windows Explorer.
2. Щелкните правой кнопкой мыши по каталогу FiveLakes, а затем выберите команду **Sharing**.
3. Перейдите на вкладку **Security**, а затем нажмите кнопку **Permissions**.
4. В списке **Name** выберите **Everyone**.
5. В списке **Type of Access** укажите **Full Control**, щелкните по кнопке **OK**, а затем вновь нажмите кнопку **OK**.

Файлы лабораторных работ

Для установки файлов лабораторных работ запустите Allfiles.exe — самораспаковывающийся архив, расположенный на CD-ROM. Следуйте соответствующим приглашениям и установкам, принятым по умолчанию.

Завершение настройки Web-сайтов

Для завершения работы над представленными в данном курсе Web-сайтами:

1. Скопируйте все файлы из каталога <папка_установки>\SampSite \FiveLakesSite в каталог C:\Inetpub\wwwroot\FiveLakes.
2. Скопируйте все файлы из каталога <папка_установки>\SampSite \RamonaSite в каталог C:\Inetpub\wwwroot\Ramona.

Проверка

В поле **Address** браузера Internet Explorer введите: **http://localhost/FiveLakes**, а затем нажмите клавишу <Enter>. Вы получите доступ к Web-сайту FiveLakes.

Повторите эту процедуру, введя адрес **http://localhost/Ramona**. Вы перейдете на сайт Ramona.

Перечень инсталлируемого программного обеспечения

При инсталляции программного обеспечения и выполнения задач, связанных с конфигурированием, соблюдайте очередность установки в соответствии с представленным ниже списком. Прочтите все подробные инструкции в руководстве по установке программного обеспечения.

- ❒ Windows NT 4.0 Server (без IIS 2.0)
- ❒ Конфигурационные драйверы
- ❒ Visual Studio 6.0 Enterprise Edition
- ❒ Windows NT Server 4.0 Service Pack 3
- ❒ Internet Explorer 4.01 SP1
- ❒ Visual Basic 6.0
- ❒ Visual InterDev 6.0 (необязательно)
- ❒ MSDN Library (необязательно)
- ❒ Windows NT 4.0 Option Pack (выборочная установка, без FrontPage Server Extensions)
- ❒ FrontPage Server Extensions — version 3.0.2.1706
- ❒ Windows NT Server 4.0 Service Pack 4
- ❒ Internet Explorer 5.0 (минимальная установка, включая Virtual Machine и Wallet)
- ❒ SQL Server 7.0 Standard Edition (типовая инсталляция)
- ❒ Установка автоматического запуска службы MSDTC
- ❒ MDAC 2.1
- ❒ Создание и настройка базы данных SQL Server
- ❒ Создание System DSN для баз данных SQL Server
- ❒ Site Server 3.0
- ❒ Site Server 3.0 Commerce Edition
- ❒ Site Server 3.0 Service Pack 2
- ❒ ADSI 2.5

- ❒ Commerce Interchange Pipeline Manager
- ❒ Windows NT Server 4.0 Service Pack 5
- ❒ Создание основы сайтов (Site Foundations)
- ❒ Формирование новых сайтов FiveLakes и Ramona
- ❒ Установка прав доступа к папке FiveLakes
- ❒ Размещение файлов лабораторных работ данного курса
- ❒ Окончательная настройка Web-сайтов
- ❒ Тестирование установок

Тестирование установок, поиск и устранение неисправностей

После выполнения установок программного обеспечения проведите следующие тесты. Если в ходе тестирования будут выявлены какие-либо проблемы, посмотрите, соответствуют ли они одному из перечисленных симптомов. Хотя эти тесты, предназначенные для выявления неисправностей, не являются исчерпывающими, они могут помочь определить наиболее типичные проблемы выполнения программ курса, вызванные нарушениями при установке.

Тестируем функциональность сайта с примерами

Данный тест проверяет функциональность Web-сайтов с примерами.

Откройте сайт **http://localhost:80/FiveLakes** в Internet Explorer и сделайте заказ на книгу. После того как этот заказ будет отправлен и вы получите номер подтверждения, запустите SQL Enterprise Manager, чтобы проверить данный заказ путем просмотра таблицы `receipt_item` сайта Ramona. Эта таблица должна содержать `SKU` и согласованную стоимость только что сделанного вами заказа.

Шаги теста

Покупка книги на сайте FiveLakes:

1. В Internet Explorer откройте **http://localhost/fivelakes**.
2. Щелкните по изображению **Computers**.
3. Выберите категорию `Computer Networks`.
4. Щелкните на изображении **BUY NOW** (Купить сейчас), чтобы внести этот продукт в покупательскую корзину.

5. Щелкните по кнопке **Continue Checkout** (Продолжить выбор) для приобретения книги.
6. На странице **Shipping** (Отгрузка) щелкните элемент **Add address**.
7. В области **Add a New Address** введите соответствующие данные в поля формы и затем щелкните кнопку **OK**.
8. На странице **Shipping** нажмите кнопку **Total** (Итого). Если появится диалоговое окно **Security Information**, щелкните по кнопке **Yes**.
9. На странице **Final Purchase Approval** щелкните **Add card**, а затем выберите **Visa** из списка.
10. В диалоговом окне **Add a New Credit Card** щелкните кнопку **Next**.
11. В диалоговом окне **Credit Card Information** введите следующую тестовую информацию:
    - укажите тип кредитной карты — `Visa`;
    - задайте номер — `4111-1111-1111-1111`;
    - щелкните кнопку **Next**.
12. В диалоговом окне **Credit Card Billing Address** из раскрывающегося списка **Billing address**, выберите адрес оплаты, а затем щелкните **Next**.
13. В диалоговом окне **Credit Card Password** в полях **Password и Confirm password** введите `VISA`, а затем щелкните кнопку **Finish**.
14. На странице **Final Purchase Approval** щелкните **Purchase**.
15. В поле **Enter Password of Credit Card** введите `VISA`, а затем щелкните **OK**.
16. На странице **Purchase Confirmation** щелкните **order number** для просмотра вашей квитанции подтверждения.

Для просмотра заказа на закупку на сайте Ramona и квитанции о подтверждении на сайте FiveLakes:

1. В меню кнопки **Start** укажите пункт **Programs**, а затем последовательно команды **Microsoft SQLServer 7.0**, **Query Analyzer** для запроса таблицы `Ramona_PO`.
2. В диалоговом окне **Connect to SQL Server** щелкните кнопку **OK**.
3. В раскрывающемся списке баз данных выберите **Ramona**, а затем введите `SELECT * FROM Ramona_PO` в окне запроса. Нажмите клавишу <F5> для выполнения этого запроса.
4. В этом же списке выберите **FiveLakes**, а затем введите `SELECT * FROM FiveLakes_POReceipt` в окне запроса. Для его выполнения нажмите клавишу <F5>.

Критерии прохождения теста

Тест считается выполненным при соблюдении следующих условий:

1. Тест может быть завершен без получения каких-либо сообщений об ошибках.
2. В результате запроса к таблице `Ramona_PO` вы видете поля **SKU**, **Name**, **Quantity**, а также **Purchase Number**.
3. В запросе `FiveLakes_POReceipt` вы увидите сгенерированные `productID`, дату и время, полученное сообщение, принятый алгоритм, а также тип документа для отправленного заказа на закупку.

Поиск и устранение неисправностей

Способы решения проблем, возникших при установке, перечислены после описания самих проблем. Первые три способа указывают, что необходимо делать, если Web-сайт выдает сообщения об ошибках в самом начале работы с ним. Два последних подходят для тех случаев, когда процесс покупки книги закончился аварийно на стадии завершения покупки.

Сайт возвращает сообщения об ошибках при попытке начать с ним работу

Симптом

Открытие Web-сайта FiveLakes вызывает следующее сообщение об ошибке:

```
Microsoft OLE DB Provider for ODBC Drivers error '80004005'
[Microsoft][ODBC SQL Server Driver][Named Pipes]Specified SQL server
not found.
/FiveLakes/i_shop.asp, line 81
```

Возможная причина

Не запущена служба MSSQL, или не задан ее автоматический запуск.

Способ устранения

Установите режим автоматического запуска при перезапуске системы для службы MSSQL. Откройте окно **Control Panel**, дважды щелкните по значку **Services**, выберите службу **MSSQL** и установите ее автоматический запуск. Шаги по выполнению этих действий описаны в *разд. "SQL Server 7.0 Standard Edition"* ранее в данном приложении.

Симптом

При открытии Web-сайта FiveLakes он возвращает следующее сообщение об ошибке:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e37'
[Microsoft][ODBC SQL Server Driver][SQL Server]Invalid object name
'FiveLakes_dept'.
/FiveLakes/Default.asp, line 34
```

Возможная причина

В конфигурации **System ODBC DSN** для FiveLakes указана база данных по умолчанию `master`, а не `FiveLakes`.

Способ устранения

В окне **Control Panel** в компоненте **ODBC Data Sources** перенастройте драйвер для работы с базой данных `FiveLakes` *(см. разд. "Создание и конфигурирование баз данных SQL Server" ранее в этом приложении).*

Симптом

При открытии Web-сайта FiveLakes он возвращает следующее сообщение об ошибке:

```
Microsoft OLE DB Provider for ODBC Drivers error '80040e4d'
[Microsoft][ODBC SQL Server Driver][SQL Server]Login failed for user
'computer name\IUSR_username'.
/FiveLakes/i_shop.asp, line 81
```

Возможная причина

System ODBC DSN сайта FiveLakes настроен на использование аутентификации для NT, а не SQL Server.

Способ устранения

В окне **Control Panel** для компонента **ODBC Data Sources** перенастройте драйвер FiveLakes для использования аутентификации SQL Server. Для этого зарегистрируйтесь под именем администратора SQL — `sa`. работы с базой данных `FiveLakes` *(см. разд. "Создание и конфигурирование баз данных SQL Server" ранее в этом приложении).*

Сайт возвращает сообщения об ошибках при попытке завершить покупку книги

Симптом

При попытке завершить покупку книги на Web-сайте выдается следующее сообщение об ошибке:

```
Commerce.MtsTxPipeline error '80070005'
[FiveLakes]Component Execution failed for component[0x1] hr:
```

```
0x80070005 ProgID: Commerce.POtoFile.1 Error opening save file
Access is denied.
/fivelakes/i_util.asp, line 87
```

Возможная причина

Для каталога c:\Inetpub\wwwroot\FiveLakes необходимо установить права доступа **Full Access** для категории пользователей **Everyone**.

Способ устранения

Установите полные права доступа к каталогу сайта FiveLakes для всех категорий пользователей. Следуйте указаниям, изложенным в *разд. "Права доступа для работы с каталогом FiveLakes"* ранее в данном приложении.

Симптом

При попытке завершить покупку книги на Web-сайте выдается следующее сообщение об ошибке:

```
Error 8004e00f
```

Возможная причина

Служба MSDTC не запущена, либо не установлен ее автоматический запуск.

Способ устранения

Установите режим автоматического запуска для службы MSDTC при загрузке системы. Откройте окно **Control Panel** и, запустив компонент **Services**, выберите службу MSDTC и установите ее автоматический запуск *(см. разд. "Создание и конфигурирование баз данных SQL Server" ранее в данном приложении)*.

ПРИЛОЖЕНИЕ С

Создание COM-объектов с помощью скрипт-языков

В данном приложении представлено сравнение скрипт-языков, построенных на основе DHTML и XML. Обсуждаются достоинства и недостатки их использования в создании объектов COM[1].

Введение

Технологии развиваются стремительно, в этом нет сомнений. Если же вам нужны веские доводы, послушайте эту историю. Как-то раз, примерно год тому назад, в театре с названием "dynamic HTML (DHTML)" состоялся дебют скромных актеров, именуемых *скриптлетами* (scriptlets). Это были HTML-страницы, которые вели себя как настоящие компоненты. Они демонстрировали свойства и методы, а также возможности по обработке событий и выдаче своих уведомляющих сообщений. Исходя из этого, можно утверждать, что они сделали явный и значительный шаг вперед в длительном процессе внедрения компонентов в Web. Именно благодаря скриптлетам написание многократно используемого HTML-кода перестало представлять собой проблему. Вы можете разместить HTML-компонент, например, таблицу данных, обладающую высокими параметрическими возможностями, и при этом использовать лишь одну строку кода для того, чтобы вставить ее в любую страницу. Пожалуй, в применении скриптлетов была лишь одна отрицательная сторона — это вопрос совместимости браузеров, поскольку их использование ограничивалось рамками Microsoft Internet Explorer 4.0 и последующих версий. Но, как сказано выше, технологии развиваются стремительными темпами. И на смену решениям, которые еще вчера представлялись революционными и злободневными, сегодня приходят более глобальные и мощные подходы. То же произошло и со скриптлетами. Прошло

---

[1] Дино Эспозито (Dino Esposito), Microsoft Corporation, ноябрь 1998 г.

лишь несколько месяцев после их возникновения (о них первоначально узнали с приходом Internet Explorer 4.0) — как им было дано более конкретное название — *динамические HTML-скриптлеты* (Dynamic HTML scriptlets). Затем их роль снизилась в связи с внезапным появлением *XML-скриптлетов* (XML scriptlets) или, как теперь принято их называть, — просто *скриптлетов.*

В результате, когда речь заходит о разработке скрипт-компонентов, предназначенных для проектов, создающихся на основе Web, мы оказываемся перед выбором из двух возможных вариантов. Первый вариант — DHTML-скриптлеты, а второй — XML-скриптлеты. Они не являются взаимоисключающими, однако каждый из них имеет собственную, независимую и четко определенную область приложения. Более того, в принципе, вы можете использовать оба подхода, как на клиентской, так и на серверной стороне Web-приложения.

Сравнение DHTML- и XML-скриптлетов

Давайте рассмотрим, почему эти технологии по сути можно считать двумя сторонами одной медали. Этой "медалью" здесь является концепция программного компонента (software component). В нашем случае представляется удачным определение программных компонентов как "независимых, программируемых, многократно используемых, не связанных с конкретным языком фрагментов кода, которые можно легко встраивать в приложения". Они обладают свойствами и методами, способны инициировать события, у них есть уникальные имена и/или идентификаторы.

И DHTML- и XML-скриптлеты несомненно отвечают данному выше определению. Различие состоит в том, что XML-скриптлеты полностью основаны на модели компонентных объектов (COM), а в DHTML-скриптлетах используется совершенно другой подход, который в чем-то имитирует функциональные возможности COM. В табл. С1 представлена сводная информация о том, как DHTML- и XML-скриптлеты реализуют распространенные функциональные возможности компонентов программного обеспечения.

Таблица С1. *Сравнение XML- и DHTML-скриптлетов*

Функциональная характеристика	DHTML-скриптлет	XML-скриптлет
Интерфейс программирования (Programming interface)	Посредством функций Microsoft JScript	Библиотека типов
Механизм создания	Функция Microsoft JScript — `public_description`	Функция `DllGetClassObject()`

Таблица С1 *(окончание)*

Функциональная характеристика	DHTML-скриптлет	XML-скриптлет
Методы и свойства	Методы функции public_description	Интерфейс автоматизации COM (COM Automation interface)
События	Окно DHTML, внешний объект	Интерфейс указателя соединения COM (COM connection pointer interface)
Язык разработки	HTML и скрипт	VBScript или JScript в XML-документе
Идентификация объекта	Имя файла и MIME-тип	ProgID и CLSID
Регистрация	Не применяется. Имя файла уникальным образом определяет скриптлет	Системный реестр (System Registry)
Размещение в качестве Web-страниц	Тег <OBJECT> и MIME-тип	Тег <OBJECT> и атрибут CLASSID
Размещение в качестве приложения	Посредством управляющего элемента Microsoft ActiveX Control	С помощью непосредственной инсталляции

Программный интерфейс, представленный в DHTML-скриптлете, — это не что иное, как коллекция методов, принадлежащих динамически заданной функции JavaScript[1]:

```
<script language="JavaScript">
public_description = new CreateScriptlet;
function CreateHotImage() {
         this.get_Property1;
         this.put_Property1;
         this.Method1;
}
```

Программный интерфейс XML-скриптлета, с другой стороны, встроен в постоянную библиотеку COM. Более того, функции сгруппированы с помощью COM-интерфейсов. Благодаря DHTML-скриптлетам вы можете получить единый, всеобъемлющий интерфейс, в то время как XML-скриптлеты не ограничены в этом смысле. Для идентификации DHTML-скриптлета достаточно присвоить ему уникальное имя файла. Регистрация не нужна. Файл

[1] В данной статье автор не приводит полные тексты обсуждаемых программных кодов. — *Ред.*

можно использовать сразу после его создания или перекачки по сети на компьютер.

Размещение скриптлетов в настольных приложениях также отражает их структурные различия. Так как DHTML-скриптлет это, в основном, — Web-страница, то для его размещения вам, как минимум, нужен элемент управления **WebBrowser**, например VB-программа. Однако более удачным решением будет использование специализированного управляющего элемента-скриптлета, который можно получить на сайте Microsoft Windows Script Technologies, **http://msdn.microsoft.com/scripting/**. Так как XML-скриптлет работает точно так же, как и COM-объект, то для работы в настольном приложении необходим только экземпляр этого объекта. В Visual Basic, например, вы можете использовать следующий код:

```
Dim o As Object
Set o = CreateObject("XML.Scriptlet")
```

Разумеется, `XML.Scriptlet` будет программным идентификатором (ProgID) данного компонента.

DHTML-скриптлеты представляют собой компоненты пользовательского интерфейса, "составленные" из всевозможных комбинаций HTML-элементов: изображений, таблиц, текста, гиперссылок и т. д. Вы можете обеспечить их взаимодействие с помощью скрипт-кода и используя объектную модель Dynamic HTML, для придания конечному компоненту необходимых интерактивных свойств.

XML-скриптлеты обеспечивают более общую инфраструктуру для компонентов, созданных на основе скриптов, где большее значение придается поведению компонента, чем пользовательскому интерфейсу.

Архитектура XML-скриптлета

Можно привести следующее краткое, но точное определение XML-скриптлета: "Компонент COM, написанный на скрипт-языке". Вы можете создавать скриптлеты с помощью таких инструментальных средств разработки, как Microsoft JScript Development Software, Microsoft Visual Basic Scripting Edition, (VBScript), или на любом другом языке, где есть синтаксический анализатор, работающий с ActiveX. Здесь возникают вопросы: "А где же волшебство?" и "Куда вы спрятали все то, что обычно относится к COM, эти фабрики классов, IUnknown и тому подобное?".

Разумеется, этому есть объяснение. Что интересно, оно вытекает из одного из основных принципов COM+: отделять поведение компонента, понятное пользователю, от более или менее затейливых деталей, лежащих в основе COM. Технология COM+ обещает существенно облегчить написание компонентов, за счет использования системного динамического модуля (system

run-time module), который обеспечит по умолчанию решение таких задач, как работа с фабриками классов (class factoring), динамическими библиотеками (DLL, Dynamic-Link Library), регистрация (registering), подсчет ссылок (reference counting), диспетчеризация (dispatching), управление соединениями (connection points handling) и запросы интерфейса (nterface queries). Другими словами, XML-скриптлеты имеют динамическую библиотеку и используют метаданные для описания объекта способом, подобным описанию COM+. Вам необходимо обеспечить следующее:

- метаданные для описания реализуемых вами интерфейсов;
- коды для реализации различных методов.

Этот принцип полностью применим к архитектуре XML-скриптлетов, представленной на рис. С1. Файл скриптлета состоит из трех главных сегментов: регистрационной информации (registration information), интерфейсного преобразования (interface mapping) и скрипт-кода. Кроме регистрационной информации, которая свойственна любому компоненту COM, остальные два имеют непосредственное отношение к метаданным и коду.

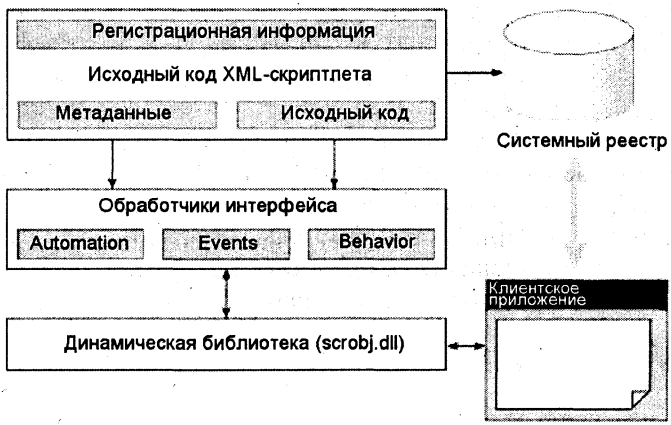


Рис. С1. Архитектура XML-скриптлетов

Клиентские приложения всегда работают с динамическим модулем скриптлета — системной динамической библиотекой scrobj.dll. Она обеспечивает необходимую инфраструктуру для того, чтобы любое клиентское приложение, работающее с COM, использовало XML-скриптлет как настоящий и эффективный объект COM. Например, динамический модуль может предоставить интерфейс `IDispatch`, с которым будут работать клиенты. Любая функция, вызванная клиентами, инициирует вызов динамического модуля со стороны соответствующего интерфейсного обработчика, для выполнения соответствующих фрагментов JScript- или VBScript-кода. Таким образом, мы действительно получаем COM-объект, созданный с помощью простейших скрипт-языков. Динамический модуль, как правило, предоставляет доступ

извне ко всем тем интерфейсам, для которых в исходном коде имеется предопределенный обработчик (interface handler). (Более подробно обработчики интерфейса мы рассмотрим позже.) В каждом XML-скриптлете всегда есть файл scrobj.dll в качестве собственного динамического серверного модуля.

XML-скриптлет представляет собой объект автоматизации (Automation object) без пользовательского интерфейса и объектной модели, лежащей в основе этого объекта. Если вам необходим пользовательский интерфейс, то следует прибегнуть к DHTML-скриптлетам.

Типичным случаем применения XML-скриптлетов является создание маленьких, эффективных и многократно используемых компонентов, запускаемых на стороне сервера и вызываемых в ASP-страницах. Далее будет рассмотрен другой, совершенно захватывающий способ использования преимуществ таких скриптлетов.

Файлы XML-скриптлета

XML-скриптлет представляет собой ASCII-файл с расширением sct. Он состоит из трех основных тегов: `<registration>`, `<implements>` и `<script>`. Первый тег содержит всю информацию, относящуюся к идентификации компонента. В частности — `ProgID` и `CLSID`. Оба этих признака являются обычными для любого объекта COM и составляют, соответственно, строку и 128-битный номер, используемые для создания экземпляра компонента. Оба они являются уникальными идентификаторами. Как правило, `CLSID` представлен в виде серий шестнадцатеричных цифр, разделенных дефисами и заключенных в фигурные скобки. При разработке XML-скриптлета в этот раздел (`registration`) вы обычно помещаете описание и номер версии. Весь исходный код XML-скриптлета расположен между парой тегов `<scriptlet>` `</scriptlet>`. Например:

```
<scriptlet>
   <registration
      Description="This is a bare-bones Scriptlet."
      ProgID="BareBones.Scriptlet"
      Version="1.00"
      ClassID="{00000000-1010-1010-83d1-f49604c10010}"
   >
   </registration>
</scriptlet>
```

Самая интересная часть рассказа начинается при рассмотрении раздела `implements`. Здесь вы определяете, какие интерфейсы будет реализовывать данный компонент. Другими словами, каждая указанная вами пара тегов `<implements>` `</implements>` соотносится с COM-интерфейсом, который клиент сможет запросить с помощью `QueryInterface`. На данном этапе этой

технологии вы не можете реализовать какой-либо COM-интерфейс с помощью скриптов, кроме тех, для которых имеется соответствующий обработчик интерфейса. Далее для некоторого упрощения, интерфейсы не идентифицируются по их CLSID-идентификаторам или программным именам (как, например, ISomething). Напротив, интерфейсы идентифицируются по атрибуту type, содержимое которого служит своего рода ключевым словом и сообщает динамическому модулю, какой COM-сервер необходимо вызвать для обработки клиентских запросов. Интерфейс может также иметь атрибут-идентификатор (id), предназначенный для единственной цели — позволить вам в программном коде обращаться с ним как с объектом. Очень удобно, когда обработчик интерфейса имеет собственную объектную модель. То, что происходит внутри элемента, представленного парой тегов <implements> </implements>, зависит от природы самого интерфейса, а также от того, насколько высоким является уровень конкретного обработчика. Ниже приведен типичный раздел implements:

```
<implements type="Automation" id="dispatcher">
   <method name="Hello">
      <parameter name=numHello />
   </method>
</implements>
```

Наконец, раздел script включает в себя весь скрипт-код, необходимый для реализации различных функций во всех интерфейсах. Например, в предыдущем скриптлете может использоваться такой раздел script, в котором метод возвращает строку из заданного количества слов "Hello":

```
<script language="JavaScript">
   function Hello (numHello) {
      var strText = "";
      for( i=0; i<numHello; i++)
         strText += strText + "Hello";
      return strText;
   }
</script>
```

Обработчики интерфейса

Обработчик интерфейса — это исполняемый COM-сервер, который обеспечивает стандартную реализацию заданного интерфейса. Это одна из наиболее важных функциональных особенностей XML-скриптлета. Иными словами, в полной коллекции обработчиков интерфейсов перечислены все COM-интерфейсы, реализованные в данном скриптлете.

На настоящий момент для таких обработчиков существует ряд ограничений. Во-первых, вы не можете задавать интерфейсы по своему выбору. Во-вто-

рых, пока нет документированного способа создания собственных обработчиков. Сегодня вы можете писать, главным образом, XML-скриптлеты, которые являются Automation-серверами. С выходом Internet Explorer 5.0, количество доступных обработчиков значительно возрастет, включая обработку событий, поведения и ASP-кода.

Automation-обработчик берет на себя заботу о Automation-интерфейсах. Обычно он реализует `IDispatchEx`. Обработчик событий (event handler) предоставляет все необходимое для инициирования событий, а именно конечные интерфейсы (connection point interfaces). Как вы можете заметить, между обработчиками и интерфейсами существуют отношения типа "один-ко-многим".

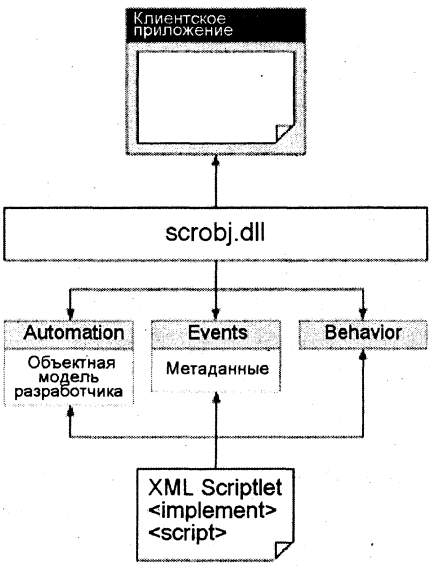


Рис. С2. Как работает XML-скриптлет

Каждый обработчик предназначен для поддержки конкретного интерфейса (или набора интерфейсов) и располагается между скриптлетом и клиентом, обеспечивая их взаимодействие, как показано на рис. С2. Из скрипт-кода вы можете, однако, "вести диалог" лишь с теми объектами, в которых объявлен (expose) Automation-интерфейс. Поэтому обработчик интерфейса должен иногда предоставлять собственную объектную модель, чтобы с ней могли работать скриптлеты. В свете вышесказанного, роль, которую играют обработчики интерфейса, можно перефразировать следующим образом: обработчик интерфейса — это объект, реализующий один и более COM-интерфейсов, плюс все то, что позволяет осуществлять их вызов и использовать в скрипт-коде. Кроме того, для любого обработчика требуются различные теги XML для представления правильной информации в разделе `implements`.

Обработчик Automation Handler

Automation Handler (обработчик автоматизации) состоит из списка свойств и методов, используемых в заданном скриптлете. Механизм диспетчеризации скрыт внутри исполняемого модуля. Допустимый синтаксис требует указания элементов с помощью тегов <method> </method> и <property> </property>. У метода должно быть имя и, возможно, внутреннее имя для идентификации скрипт-процедуры, с помощью которой реализован данный метод. Если вы опустите внутреннее имя, то будет считаться, что процедура имеет то же имя, что и метод. Свойства могут быть представлены как простые глобальные переменные, или же вы можете определить специальные процедуры для записи и считывания их значений. Таким образом, существует возможность сделать их доступными только для записи или только для чтения.

```
<method name="Method1" internalName="DoMethod1" />
<property name="Prop1" internalName="m_Prop1" />
```

В предыдущем примере показано объявление метода (Method1), встроенного в виде кода в функцию DoMethod1, и свойство Prop1, представленное переменной m_Prop1. Атрибут internalName необязателен. Переменная m_Prop1 является глобальной переменной, действующей как контейнер для значений свойства. В качестве альтернативы вы могли бы иметь примерно следующее:

```
<property name="Prop1">
<get internalName="DoGetProp1" />
<put />
</property>
```

где свойство Prop1 считывается с помощью функции DoGetProp1, а запись осуществляется с использованием функции put_Prop1, отвечающей соглашению по стандартному наименованию. Согласно этим правилам, считывание должно производиться с помощью процедуры, имя которой начинается с get_, а далее следует public-имя считываемого свойства. Разумеется, все необходимые функции, как для методов, так и для свойств, должны быть реализованы в разделе script.

Обратите внимание, что такое определение методов и свойств является способом высокого уровня по отношению к Automation. Все COM-объекты, необходимые на этом уровне и ниже — до интерфейсов диспетчеризации, — выполняются с помощью данного обработчика.

Точки соединения

Если вы хотите, чтобы ваш скриптлет запускал события, вам нужно перенести их в обработчик событий, который обеспечивает необходимую инфраструктуру COM на основе стандартной реализации интерфейсов

`IConnectionPoint` и `IConnectionPointContainer`. Все, что я говорил прежде о необходимости дополнительного уровня в коде Automation, между скриптлетом и обработчиком, обычно также имеет силу и для событий. Обработчик событий предназначен для обеспечения сервиса — запуска событий из скрипт-кода. По своей идее, это простое действие, однако оно требует от обработчика определенной внутренней работы. Ему необходимо предоставить исходящий интерфейс, позволить клиентам подключиться (sink), и затем найти зарегистрированные подключения при выдаче заданного уведомления. Этот процесс следует пояснить. Фактически, объект обработчика событий предоставляет метод `fireEvent`, который позволяет скриптлетам уведомлять о своих событиях внешний мир:

```
<implements type="Event" id="MyScriptlet">
   <event name="BeginWork" />
   <event name="WorkCompleted" />
</implements>
```

В этом примере демонстрируется скриптлет, представляющий два события: `BeginWork` (начало работы) и `WorkCompleted` (работа выполнена). COM всегда ассоциирует событие с идентификатором диспетчера (dispatch identifier), именуемого `dispid`. Этот идентификатор затем компилируется в библиотеку типов и используется клиентом для связи с событием. Обычно данный модуль служит для генерации библиотеки типов (type library) для того, чтобы скриптлет (смотрите следующий пример) мог выдать автоматические идентификаторы диспетчера в представленные события. Однако, добавив атрибут `dispid`, вы можете сами решить, какой именно идентификатор будет обозначать данное событие.

```
<event name="BeginWork" dispid=65 />
```

Вот пример того, как происходит инициирование событий из скрипт-кода:

```
<script language="JScript">
   function DoWork() {
      MyScriptlet.fireEvent("BeginWork")
      // здесь должен быть код
      MyScriptlet.fireEvent("WorkCompleted")
   }
</script>
```

Если бы мы использовали атрибут `default` в обработчике событий, то в данном скриптлете могли бы обойтись без префикса `MyScriptlet`.

Обработчики Behaviors в Internet Explorer 5.0

Третьим типом обработчиков интерфейсов, существующим на сегодняшний момент времени, являются так называемые обработчики поведения (behaviors) Internet Explorer 5.0.

Насколько вы можете судить, они поддерживаются в Internet Explorer лишь с версии 5.0. Кроме того, обработчики поведения — это не что иное, как XML-скриптлеты, которые реализуют семейство определенных интерфейсов Internet Explorer 5.0, таких как `IElementBehaviorXxx`. Идея, лежащая в основе этих обработчиков, довольно проста. Они предназначены для сокращения скрипт-кода, осуществляющего подготовку тегов HTML к каскадным таблицам стилей (CSS). Таким образом, когда вы определяете стиль элемента или группы элементов, вы можете указать некое "поведение" точно так же, как графические атрибуты — шрифт и цвета.

Поведение — это компонент, который определяет, что могут делать элементы некоторого класса CSS. Например, если вам необходимо, чтобы они обрабатывали события DHTML или предоставляли дополнительные методы или свойства, вы можете написать обработчик Behavior. К тому же использование этих обработчиков делает Web-страницу гораздо более удобной для чтения и упрощает ее управление путем значительного сокращения потребности в скрипт-коде на сайте. В то же время, это вовсе не ограничивает возможностей усиливать элементы HTML — будь то тег `<img>` или же какой-нибудь "заказной" элемент, например, DHTML-скриптлет — дополнительными функциональными возможностями.

Для реализации характера поведения необходим механизм Automation для правильной обработки специальных элементов HTML-кода, в котором используется данный элемент. Например, предположим, что у нас есть "сценарий поведения" тега `<img>` и мы хотим, чтобы он осуществлял вывод различных изображений, в соответствии со значением атрибута. Этот атрибут может указывать, скажем, глубину цвета. На HTML-странице код, предназначенный для этого, может выглядеть так:

```
<img id="one" class="colorimg" src="" color="16" child="one">
```

Внутри класса `colorimg` могут быть такие определения:

```
<style>
  .colorimg {behavior: url(colorimg.sct)}
</style>
```

Скриптлет colorimg.sct содержит информацию о характере поведения всех элементов класса `colorimg`. Код такого скриптлета будет выглядеть следующим образом:

```
<scriptlet>
<implements type="behavior" />
<implements type="automation" />
  <property name="color" />
  <property name="child" />
</implements>
<script language="JavaScript">
```

```
attachEvent( "onload", event_onload);
function event_onload() {
  document.all(child).src = child + color +".gif";
}
</script>
</scriptlet>
```

Этот скриптлет обеспечивает поведение, а также Automation-интерфейс, с помощью двух свойств: `color` и `child`. Данный скриптлет запускается при возникновении события `onload` той Web-страницы, которая вызывает указанный скриптлет. Это выполняется при помощи функции `attachEvent`, которая является специфической для объектной модели DHTML Internet Explorer 5.0. Реагируя на это событие, скриптлет "решает", какое именно изображение будет выводить тег `<img>`. Имя изображения задано именем тега, переданным свойством `child`, и количеством цветов — через свойство `color`.

```
<img id="one" class="colorimg" src="" color="16" child="one">
```

В данном случае именем файла с изображением будет one16.gif.

Скриптлеты и клиентские приложения

Так как XML-скриптлеты действительно являются COM-объектами, вы можете их использовать в любой среде разработки, согласующейся с COM, в качестве которой, ввиду нейтральности COM к языку, может выступать любое средство — от Visual Basic до Delphi, и от Microsoft Visual C++ до ASP-кода.

Давайте бегло рассмотрим, каким образом происходит взаимодействие между XML-скриптлетами и клиентскими приложениями. Для создания экземпляра XML-скриптлета клиентское приложение не использует какой-либо подход, который бы отличался от тех, что применяются для остальных COM-объектов. Так, в Visual Basic вы можете создать объект следующим образом:

```
Dim o As Object
Set o = CreateObject("BareBones.Scriptlet")
o.Hello 4
Set o = Nothing
```

В основе метода `CreateObject` (или его аналога в других языках) лежит обычный диалог между клиентами и COM-серверами. В предыдущем фрагменте кода клиент запросил интерфейс `IDispatch` и метод `Hello`. Файл, который вызывается после `CreateObject` — это динамический модуль scrobj.dll. Он работает как модуль-посредник (proxy) между клиентом и обработчиком интерфейса Automation и возвращает клиенту ссылку на интерфейсы, ис-

пользуемые в данном скриптлете, но не реализует их, а "полагается" на специализированные модули, именуемые обработчиками интерфейса.

В этом коде на Visual Basic мы применили метод `CreateObject` и позднее связывание (late binding). Если у вас есть библиотека типов для скриптлета, вы можете также использовать раннее связывание.

Библиотека типов скриптлета

Библиотека типов представляет собой файл, в котором сохраняется детальная информация об интерфейсах, реализуемых COM-компонентом. XML-скриптлеты не являются исключением. Несмотря на то, что создание библиотеки типов не является строго необходимым, эта библиотека может оказаться очень полезной, особенно при использовании компонентов в Visual Basic. Например, такая технология, как Microsoft IntelliSense в значительной мере основана на библиотеках типов, что позволяет показать все методы, доступные для заданного объекта. Говоря о Visual Basic, необходимо отметить, что с помощью ссылок на библиотеку типов скриптлета, вы можете объявлять ее с помощью квалификатора `WithEvents` и обеспечивать захват указанных в ней событий.

Существует два способа генерирования библиотеки типов для XML-скриптлета. Вы можете это сделать в режиме off-line, щелкнув правой кнопкой мыши по sct-файлу и выбрав необходимую команду. В качестве альтернативы вы также можете сделать это программным путем:

```
Set oTL = CreateObject("Scriptlet.GenerateTypeLib")
oTL.AddURL "c:\MyDir\myScriptlet.sct"
oTL.Write
oTL.Reset
```

Этот код может пригодиться где угодно — в другом компоненте, предназначенном для генерации библиотек типов, или даже в самом скриптлете, для однократного выполнения на стадии регистрации. Если вы хотите запустить код при регистрации скриптлета, можете включить тег `<script>` в элемент `registration` и определить функцию `register`:

```
<registration
   Description="This is a bare-bones Scriptlet."
   ProgID="BareBones.Scriptlet"
   Version="1.00"
   ClassID="{00000000-1010-1010-83d1-f49604c10010}"
>
   <script language="VBScript">
      Function register()
      ' создает библиотеку типов
      End Function
```

```
      Function unregister()
      ' делает то, что вам необходимо
      End Function
   </script>
</registration>
```

Функция `unregister` выполняется во время исключения из реестра (unregistering).

Библиотека типов может включать в себя более одного скриптлета. В этом случае повторно вызывайте `AddURL` с различными именами файлов. Для получения более детальной информации об опциях, используемых при генерации библиотеки типов, обратитесь к документации по скриптлетам на **http://msdn.microsoft.com/scripting/**.

XML-скриптлеты и Windows Scripting Host

XML-скриптлет является COM-объектом, состоящим из XML и скрипт-кода, работающим на верхнем уровне динамического окружения. Динамическая библиотека берет на себя заботу о тех чудесах, которые позволяют скриптлету вести себя по отношению к клиенту, как настоящий COM-объект, в том же стиле, как работают Microsoft Visual J++, Microsoft Visual FoxPro и Microsoft Visual Basic. Мы можем запускать XML-скриптлеты в качестве "безмолвных" модулей без пользовательского интерфейса, как на клиентской (компоненты DHTML и обработчики поведения (behaviors)), а также на серверной стороне (ASP-страницы). Кроме того, на клиентской стороне мы можем использовать вариант, именуемый DHTML-скриптлетами, представляющими собой программируемые и встраиваемые DHTML-страницы.

Появление Microsoft Windows Scripting Host (WSH) открыло новую впечатляющую область для XML-скриптлетов. Если вы пишете WSH-апплеты, то обычно используете VBScript или JScript. Это быстро, просто и гораздо мощнее, чем применение пакетных файлов. А как насчет многократного использования? Хорошим ответом на этот вопрос являются XML-скриптлеты.

Давайте рассмотрим пример, который предоставляет возможность посмотреть XML-скриптлеты в действии. Наша задача заключается в том, чтобы пересчитать все файлы в заданной папке, которая последний раз модифицировалась в определенный день. Хотя это не очень трудная задача для того, чтобы решить ее на C и C++, она все же представляет несколько большую сложность для решения с помощью скриптов. Важно, чтобы код, который был однажды написан, можно было бы использовать многократно. Пусть первый пример XML-скриптлета, который позволяет это делать, называется

filefinder.sct. Программный идентификатор (progID) — `FileFinder`, скриптлет Automation представляет два метода:

```
FindFirstFileByDate(pathSpec, date)
FindNextFileByDate()
```

Вы указываете имя папки, а также строку, содержащую дату и функции, которые будут осуществлять подсчет всех найденных файлов, модифицированных в этот день. Для подсчета содержимого папки мы воспользуемся новыми объектами-оболочками, в которых допускается использование скриптов. Эти объекты были представлены в Internet Explorer 4.0:

```
Set m_Shell = CreateObject("Shell.Application")
Set m_Folder = m_Shell.NameSpace(m_PathSpec)
```

Теперь объект `m_Folder` содержит коллекцию всех элементов папки. Остается лишь сравнить дату их модификации с имеющейся у нас датой. Метод `NameSpace` возвращает объект типа `Folder`, представляющий собой коллекцию объектов `FolderItem`. Описание модели объекта-оболочки представлено в справочных файлах Internet Client SDK.

В нашем примере осуществляется последовательное считывание списка с проверкой даты. Сравнение выполняется с помощью функции VBScript — `DateDiff`:

```
FileDate = m_Folder.Items.Item(m_Index).ModifyDate
d = DateDiff("d", FileDate, m_Date)
```

Функция `DateDiff` возвращает разницу между двумя датами, выраженную в количестве дней. Такая мера измерения времени задается с помощью аргумента `"d"`. Данный скриптлет вначале осуществляет инициализацию объекта папки, а затем начинает перемещать индекс по всему списку, отыскивая все те файлы, дата изменения которых соответствует указанным требованиям. Далее необходимо вызвать функцию `FindFirstFileByDate` для инициализации механизма и получить имя первого файла. Затем — функцию `FindNextFileByDate` для всех файлов, удовлетворяющих условию поиска. Просматривая код этого примера, обратите внимание на странную строку, расположенную в начале файла:

```
<?scriptlet validate="true" error="true" debug="true"?>
```

Она служит лишь для отладки. В случае обнаружения ошибок динамический механизм скриптов выдаст оперативное сообщение.

Еще, что может вызвать ваш интерес, — это выражение `<![CDATA[ ... ]]>`, заключающее в квадратные скобки весь скрипт-код. Как будет представлено далее, наличие этой строки гарантирует, что любые возможные пропуски символов в скрипт-коде не вызовут нарушения работы XML-анализатора.

Использование этих квадратных скобок предохраняет вас от раздражающих и необъяснимых ошибок. Без этих скобок я даже получу сообщение об ошибке в комментариях! В следующем разделе объясняется, почему так происходит. Пока же имейте в виду, что тег `<CDATA>` должен быть тесно привязан к тегу `<?XML version="1.0"?>`, который расположен в самом начале XML-файла.

Второй пример демонстрирует, как использовать объект `FileFinder` в WSH-скрипте. Простой код, написанный на VBScript, принимает в режиме командной строки два аргумента, необходимых для выполнения работы: путь папки и дату. Значениями по умолчанию для данных аргументов являются диск C: и текущая дата.

Как показывает компонент `FileFinder`, с помощью XML-скриптов вы получаете способ для написания многократно используемого кода — столь же простого, как и пакетные файлы.

Соответствие XML

На сегодняшний момент мы знаем, что XML-скриптлет является главным образом XML-файлом, но как же быть с согласованностью по языку? XML представляет собой более строгий язык, чем HTML. Известно, что при написании скриптлетов наиболее распространенные ошибки — это:

- использование неверных регистров клавиатуры в тегах;
- использование специальных символов XML, например, < и > для других целей (скажем, в скрипт-коде);
- пропуск кавычек при описании содержимого атрибутов тега.

Все эти ошибки никогда не встречаются в HTML. По умолчанию динамический модуль осуществляет синтаксический анализ файла XML-скриптлета так, как если бы это был HTML-файл. Иными словами, он позволяет вам следовать предыдущим трем соглашениям, которые являются нормальными для HTML, но не для XML. С другой стороны такой свободный синтаксис, безусловно, создаст проблемы, как только вы поместите ваш код в XML-редактор, совместимый с XML 1.0. Недавно я проверил это при помощи объектной модели XML, встроенной в Internet Explorer 4.0. Я написал управляющий элемент ActiveX, работающий как "родовой" (generic) анализатор XML, переводящий содержимое XML-файла в древовидную структуру. (Кстати, его можно получить по адресу: **http:// www.microsoft.com/mind/** с исходным кодом на сентябрь 1998 г.) В нем используется компонент, который строго соответствует стандарту 1.0. В результате было невозможно работать со многими моими скриптлетами. Излишне говорить, что все эти скриптлеты прекрасно функционировали в динамическом режиме. Во всех

этих случаях в тегах скрипта содержались символы операторов "бояьше" (>) или "меньше" (<). Несмотря на то, что им предшествовали escape-символы, работа синтаксического анализатора нарушалась, а указанные символы воспринимались или как открывающие, либо как закрывающие теги, или же открывался новый тег с непредсказуемыми результатами.

Для усиления контроля на этапе выполнения и избежания слишком свободных синтаксических правил, свойственных HTML, вы можете разместить вначале скриптлета следующую строку:

```
<?XML version="1.0"?>
```

Содержимое атрибута `version` является номером версии стандарта XML, которому, согласно данному объявлению, должен соответствовать представленный XML-код. В случае возникновения ошибок объект XML-скриптлета не будет создаваться. Почти все фрагменты предыдущего кода не совместимы со стандартом 1.0. Я преднамеренно это сделал, чтобы продемонстрировать, где могут находиться ошибки. Причины, по которым данный код не совместим со стандартом, заключаются в том, что в нем нарушено одно из двух правил:

- все теги и имена атрибутов должны быть приведены в нижнем регистре;
- содержимое всех атрибутов должно заключаться в двойные или одинарные кавычки.

Внимательное отношение к стандарту требует, чтобы в скрипт-коде перед символами < и > использовались escape-символы, даже если они (< и >) присутствуют в строках комментариев. Для того, чтобы обеспечить соответствие стандарту 1.0 и в то же время оградить себя от использования escape-символов, поместите весь скрипт-код в раздел `<![CDATA[...]]>`. Квадратные скобки охватывают код функции. Например:

```
<script language="VBScript">
   <![CDATA[
   Function method()
   ' здесь должен быть скрипт-код
   End Function
]]>
```

Данное решение не зависит от используемого скрипт-языка, таким образом, вы можете применять его в скриптах JScript.

Выводы

XML-скриптлеты ушли далеко вперед по сравнению с DHTML-скриптлетами, которые первыми появились на сцене. Однако и те и другие могут

быть действительно полезны для разработчиков в тех приложениях, где они более всего подходят по своим функциональным возможностям. Так, вы можете использовать на Web-страницах DHTML-скриптлеты, если вам необходима независимость объектов пользовательского интерфейса. (Вы можете также предусмотреть обработчики поведения DHTML для этого.) Если же вам нужны рабочие объекты с дружественным программируемым интерфейсом, которые могли бы использоваться где угодно, от ASP до HTML, и от настольных приложений до WSH, тогда существует только один ответ: XML-скриптлеты.

ПРИЛОЖЕНИЕ D

Ссылки на официальные документы и материалы

Для получения информации о Microsoft Site Server Commerce Edition:

- Site Server Commerce Edition
  http://www.microsoft.com/siteserver/commerce/
- CIP Manager
  http://www.microsoft.com/siteserver/commerce/DeployAdmin/Pipeline.htm

Для изучения рекомендаций консорциума World Wide Web Consortium (W3C) по XML:

- Extensible Markup Language (XML) 1.0 Recommendation
  http://www.w3.org/TR/xsl/
- Пространство имен (namespaces) в XML
  http://www.w3.org/TR/REC-xml-names/
- XML Schema Part 1: Structures (структуры)
  http://www.w3.org/TR/xmlschema-1/
- XML Schema Part 2: Datatypes (типы данных)
  http://www.w3.org/TR/xmlschema-2/
- Extensible Stylesheet Language (XSL)
  http://www.w3.org/Style/XSL/
- XSL Transformations (XSLT) Specification Version 1.0
  http://www.w3.org/TR/1999/WD-xslt-19990421.html
- Associating Style Sheets with XML documents Version 1.0
  http://www.w3.org/TR/xml-stylesheet/

Материалы по XML:

- MSDN Online: XML Developer Center
  http://msdn.microsoft.com/xml/default.asp
- OASIS: Organization for the Advancement of Structured Information Standards
  http://www.oasis-open.org/
- XML.com
  http://www.xml.com/
- XMLINFO: The XML Information Site
  http://xmlinfo.com/
- XMLephant. The BIG XML Resource
  http://xmlephant.com/
- VBXML: An ASP and VB developer's home for XML
  http://www.vbxml.com
- TIBCO Extensibilty. XML Infrastructure Solutions
  http://www.extensibility.com/
- STG XML. Validation Form
  http://www.stg.brown.edu/service/xmlvalid/
- DTDGenerator Frontend
  - **http://pault.com/Xmltube/dtdgen.html**

ПРИЛОЖЕНИЕ E

Описание компакт-диска

На компакт-диске содержатся следующие файлы и папки:

- Файл autorun.exe. Когда компакт-диск установлен в дисковод CD-ROM, или же когда вы дважды щелкаете мышью по имени этого файла, указанная программа открывает диалоговое окно и позволяет просмотреть содержимое компакт-диска или установить Internet Explorer.
- Файл default.htm предоставляет вам информацию о ресурсах, относящихся к данному курсу.
- Файл readme.txt содержит описание (в формате ASCII) содержимого компакт-диска.
- Папка DemoCode включает в себя демонстрационный код. Если для какого-либо раздела файлы с демонстрационным кодом не предусмотрены, тогда соответствующий каталог в папке DemoCode отсутствует.
- Папка Htm содержит Web-страницы с кратким описанием лабораторных работ.
- Папка Ie5 содержит дистрибутив Internet Explorer 5.0.
- Папка Labs включает файлы, используемые в лабораторных работах.
- Папка Menu заключает в себе элементы для autorun.exe.
- Папка Practices содержит файлы, используемые в практических работах. Если для какого-либо раздела практических работ нет, то тогда отсутствует и соответствующий каталог в папке Practices.
- Папка SampSite содержит файлы сайта с примерами, относящимися к данному курсу.
- Папка Setup включает в себя самораспаковывающиеся архивы для инсталляции файлов, необходимых к данному курсу.

Ответы на вопросы

Глава 1

1. Достоинствами электронной коммерции данного класса являются:
   - низкие закупочные цены;
   - малая продолжительность цикла;
   - эффективный сервис;
   - низкие цены и издержки сбыта;
   - новая конъюнктура рынка.
2. Используются следующие протоколы:
   - SSL;
   - SET.
3. Эффективный Web-сайт электронной коммерции класса business-to-busines должен обеспечивать:
   - электронные каталоги;
   - доступ с соблюдением безопасности;
   - решения, основанные на стандартах;
   - более простой и дешевый обмен документами.

Глава 2

1. Каталог позволяет вам распределенную в широких пределах информацию в едином и доступном месте. Важная информация, содержащаяся в документах каталога, извлечена и проиндексирована. Когда пользователь ищет

эту информацию, программа Search производит выборку всех документов в каталоге, содержащих соответствующие ключевые слова.

2. Поисковая страница включается в состав сайта при помощи ссылки на нее, созданной на сайте.
3. В зависимости от типа формируемого каталога, определение каталога должно содержать:
   - имя каталога;
   - начальный адрес и стратегию обхода;
   - правило, определяющее сайт и путь;
   - типы файлов;
   - список распространения;
   - установки журнала ошибок сбора информации;
   - расписание формирования каталога;
   - распределение доступа и отображения.

Глава 3

1. В CPP-конвейере выполняются OPP-компоненты, которые производят вычисления итоговой стоимости заявки. В процессе этих вычислений учитываются такие факторы, как скидки (discounts), налоги (taxes), а также транспортные расходы (shipping charges).
2. CPS-конвейер проверяет действительность заявки, передает и записывает заявки по заказам (order requisitions) на закупку в базу данных.
3. Данный бизнес-объект объявляется под именем `Items`.
4. Могут использоваться компоненты:
   - `PipetoPipeTransfer`;
   - `SendSMTP`;
   - `SendHTTP`;
   - `SendDCOM`.
5. Применяются следующие CI-конвейеры:
   - Transmit pipeline (CIT-конвейер);
   - Receive pipeline (CIR-конвейер).
6. Вы можете использовать сертификаты в ступенях добавления цифровой подписи (Digital Sign) и шифрования (Encrypt).

Глава 4

1. Для вызова внешних скриптов используется компонент `Scriptor`.
2. В процессе работы CIP-конвейера необходимо передавать в качестве параметров транспортный и контекстный словарь.
3. Цифровые подписи используются в ступенях дешифрования (Decrypt stage) и проверки цифровой подписи (Verify Digital Signature stage).
4. Компонент `OpenHeader` создает комбинированное сообщение лишь тогда, когда отправитель запрашивает квитанцию подтверждения.

Глава 5

1. Такими элементами являются исходная организация (home organization), наборы параметров документа (document profiles), адреса приема данных (receive locations), наборы коммерческих параметров (trading profiles) и соглашения (agreements).
2. Набор коммерческих параметров (Trading Profile) содержит информацию об исходной организации (home information), наборы параметров документа (document profiles), информацию о сертификации (certificate information), и информацию о местоположении (адрес) (location information).
3. Методами, которые связаны с объектами `SendReceive`, являются `Send` `ReceiveStandard` и `ReceiveCustom`.
4. Идентификатор категории (category ID) reg-файла указывает ступень конвейера, в которой используется данный компонент.

Глава 6

1. Объекты данных должны быть правильными (well-formed) и могут быть допустимыми (valid).
2. В DTD указывается, какие элементы должны быть включены в XML-документ и содержат ли они обязательные данные. Если DTD обнаружит отсутствие данных, оно сообщит об этом разработчику как об ошибке[1].
3. Различие, существующее между HTML и XML, показано в табл. 6.1.
4. Разработка схем XML-данных (XML-Data Schemas) была обусловлена следующими ограничениями DTD:
   - DTD не являются XML-документами, а написаны на другом языке;

---

[1] Имеется в виду, что XML-процессор обнаружит это несоответствие и выдаст соответствующее сообщение. — *Пер.*

- DTD не предоставляет возможности контроля над типами данных, принимаемых элементами.

Глава 7

1. XSL специально разработан в соответствии с требованиями XML. Поскольку все таблицы стилей XSL являются действительными XML-документами, пользователи могут работать с ними как с разметкой. К тому же XSL поддерживает языки скриптов и поэтому является расширяемым.
2. Действительными узлами на исходном дереве являются: корень (root), элемент (element), текст (text), атрибут (attribute), пространство имени (namespace), инструкция по обработке (processing instruction) и узлы комментариев (comment nodes).
3. Пространством имени, принятым по умолчанию для всех таблиц стилей XSL, является **http://www.w3c.org/TR/WD-xsl**.
4. XSL по умолчанию сохраняет пробелы подобно XML.

Глава 8

1. XML-скриптлеты должны иметь теги `<registration>`, `<implements>` и `<script>`.
2. XML позволяет писать COM-объекты на таких скрипт-языках, как Microsoft JScript, Microsoft Visual Basic Scripting Edition или Perl, а также представлять все элементы описания COM-объекта как метаданные. Это значительно упрощает работу по программированию COM-объектов.
3. Таким динамическим файлом, используемым по умолчанию XML-скриптлетами, является файл scrobj.dll.
4. DHTML-скриптлеты представляют интерфейсы программирования при помощи функции Microsoft JScript, в то время как XML-скриптлеты используют библиотеки типа (type libraries).

Глава 9

1. Электронный конверт содержит одну и более функциональных групп, каждая из которых включает один и более наборов транзакции.
2. В основном потоке транзакции EDI замещает бумажные документы их электронными эквивалентами. Таким образом, EDI способствует более

быстрому продвижению продукции на рынок сбыта. Кроме того, EDI снижает издержки на почтовые расходы и повышает надежность данных.

3. Хотя EDI предлагает значительные преимущества, не все компании используют эту технологию. Следовательно, когда та компания, в которой реализован электронный обмен документами (EDI), взаимодействует с той, где он не используется, возникает необходимость в преобразовании в формат EDI и из него.

4. Таблица стилей XSL может лишь манипулировать XML-документами. Так как наборы транзакций EDI не являются XML-документами, они не могут быть преобразованы в другие форматы с помощью таблиц стилей XSL.

Глава 10

1. Сеанс создается, когда пользователь запрашивает ASP-страницу.
2. Данные о сеансе хранятся в Membership Directory как динамические данные. Они не записываются на диск.
3. Данные о пользователе могут храниться в одном и более информационных массивах (data stores). AUO может осуществлять доступ к источнику данных и представлять виртуальную схему, которая доступна для приложений. Далее приложениям не нужно осуществлять доступ к каким-либо источникам для получения информации о пользователе.

Глава 11

1. Необходим цифровой сертификат, который представляет собой текстовый файл, содержащий информацию о владельце сайта, сертифицирующем центре, а также шифрованная информация для идентификации этого центра.
2. На стадии авторизации на кредитной карте покупателя сохраняется некий резервный запас, но эти фонды (средства) не переводятся со счета покупателя на торговый счет. Этап приема транзакций вступает в силу после того, как происходит доставка товаров. На этом этапе осуществляется перевод фондов (средств) на торговый счет.
3. Необходимо включить компонент, содержащийся в программном обеспечении обработки платежей, в конвейер Microsoft Commerce Pipeline с помощью программы Microsoft Commerce Pipeline Editor.
4. Microsoft Passport хранит конфиденциальную информацию в виде зашифрованных данных и отправляет эти данные в соответствии с прото-

колом SSL. Кроме того, когда пользователь выходит с сайта, все cookies, относящиеся к Microsoft Passport, удаляются.

Глава 12

1. Новый каталог может быть расположен где угодно на хост-компьютере, за исключением той папки, в которой содержатся файлы существующего сайта.
2. Нет. Вы можете лишь добавить и модифицировать варианты продукта на настраиваемом сайте.
3. Объект `NewMail` помогает создавать и отправлять сообщения электронной почты из приложений Microsoft Visual Basic, Microsoft Visual C++ или VBScript.

Глоссарий

Access control (управление доступом) — предоставление или отказ в праве доступа пользователей на просмотр и/или изменение каталогов, содержащих файлы и другие объекты.

Active Directory Service (ADS, служба активного каталога) — стандарт, облегчающий компаниям и разработчикам доступ и управление службами каталогов, такими как, например, каталоги NetWare 3.x и 4.x и любые другие LDAP-совместимые каталоги, с помощью широко распространенного API-интерфейса.

Active Directory Service Interfaces (ADSI, интерфейсы службы активного каталога) — компонент *интерфейсов открытой службы каталогов* (ODSI, Open Directory Service Interfaces) *открытой системной архитектуры Windows* (WOSA Windows Open System Architecture). Служба каталога является частью распределенной вычислительной среды. Эта служба обеспечивает способ нахождения и идентификации пользователей и ресурсов в системе. Единая связанная система может состоять из различных сетевых сред, каждая из которых предлагает разные службы каталогов. ADSI определяет общий набор интерфейсов, который заключает в себе те свойства и методы объектов служб каталогов, которые являются общими для разнотипных сред, включая LDAP-совместимые службы, такие как каталог принадлежности (Membership Directory) и NTDS.

Active Server Pages (ASP, активные серверные страницы) — среда для скриптов на стороне сервера, которая обычно используется для создания интерактивных высокопроизводительных приложений Web-сервера. Выполнение ASP-скриптов производится в большей мере на стороне сервера, чем на стороне клиента. Это гарантирует осуществление сервером всей работы, связанной с генерацией отправляемых в браузеры HTML-страниц.

Active User Object (AUO, объект активного пользователя) — COM-объект, именуемый Membership.userobjects, который собирает в единое целое свойства пользователя из различных механизмов хранения свойств служб

активного каталога (ADS, Active Directory Service) в единую унифицированную коллекцию (пространство имен). Приложения могут получать доступ к свойствам пользователя с помощью AUO. При этом приложениям не нужно "знать" ни истинное место хранения этих свойств, ни механизм идентификации пользователя. AUO используется в ASP-файле, как если бы это был простой ADS-провайдер, и обеспечивает прозрачное управление поиском объекта пользователя у множества провайдеров.

ActiveX — технологии Microsoft, которые являются открытой платформой, расширяющей архитектуру Windows за счет включения аспектов и возможностей Internet и intranet. ActiveX охватывает Java- и COM-технологии.

ActiveX control (элемент управления ActiveX) — встраиваемый элемент управления, написанный в соответствии со спецификацией ActiveX. Он может быть встроен в форму Microsoft Visual Basic, ресурс Microsoft Visual C++ или в HTML-страницу.

ActiveX Data Objects (ADO, объекты данных ActiveX) — объекты, которые обеспечивают доступ к данным, хранящимся в базах данных, отвечающих технологии ODBC, и которые могут использоваться для отображения и обработки на Web-страницах. Подробная информация об этих объектах содержится в документации Visual Basic.

ActiveX server component (серверный компонент ActiveX) — обычно, динамически компонуемая (dynamic-link) библиотека, предоставляющая свои функциональные возможности для активных серверных страниц (ASP) через определенные и реализованные в библиотеке методы. В отличие от типичных объектов Automation, управляющий компонент ActiveX зарегистрирован операционной системой таким образом, чтобы его можно было использовать в ASP. См. также *Active Server Pages*.

ADS — см. *Active Directory Service*.

ADS provider (ADS-провайдер) — программное обеспечение, которое реализует интерфейсы службы активного каталога (ADSI, Active Directory Service Interfaces), их методы и свойства. Провайдер может поддерживать также дополнительные интерфейсы, методы и свойства, чтобы клиентские ADSI-приложения могли использовать все свойства обычной службы каталога.

ADSI — см. *Active Directory Service Interfaces*.

Application (приложение) — программа или некоторое множество Web-страниц, выполняющих определенную задачу.

Area (область) — раздел Web-сайта, обеспечивающий функциональность путем адресования одного и более бизнес-приложений или нужд. Область обычно состоит из нескольких HTML-страниц.

ASP — см. *Active Server Pages.*

Attribute (атрибут) — пара "имя-значение", размещенная внутри отмеченного тегами элемента, модифицирующая некоторые его свойства.

AUO — см. *Active User Object.*

AUOconfig file — ASP-файл, ассоциированный с Membership Server, который используется совместно с реестром для обеспечения конфигурации AUO.

AUO-provider (AUO-провайдер) — объект-контейнер ADSI, используемый для содержания свойств пользователя, доступный для объекта активного пользователя (AUO, Active User Object). В качестве AUO-провайдера может быть любой контейнер службы каталога.

Authentication (аутентификация) — процесс верификации регистрационной информации пользователя.

Authentication mode (режим аутентификации) — соглашение, по которому аутентифицируются пользователи, службы и приложения, определяемое местом хранения данных о полномочиях (имен и паролей). Сайты на базе Site Server могут использовать либо Membership Authentication (данные о полномочиях хранятся в Membership Directory), либо Windows NT Authentication (данные о полномочиях хранятся в базе данных каталога Windows NT Server).

Authorization (авторизация) — процесс предоставления или запрещения доступа к каталогам или другим объектам, защищенным средствами управления доступом.

Automatic approval (автоматическое подтверждение) — процесс регистрации, проходящий в онлайновом режиме и не требующий вмешательства администратора.

Bind (связывать) — идентифицировать себя как пользователя или зарегистрироваться в LDAP Service.

Bind as — регистрироваться в качестве отдельного пользователя.

Branch — узел или часть области (subarea) на Web-сайте.

Branded icons — уникальные значки или логотипы, отображаемые в браузерах клиентов, служащие для идентификации провайдера канала и содержимого его разделов.

Broken link (разрушенная ссылка) — ссылка на ресурс, который не может быть найден анализатором содержимого (Content Analyzer), т. к. URL не действителен, ресурс, на который указывает ссылка, не существует, или же сервер, содержащий ресурс, занят или неисправен.

Buy Now control (элемент управления Buy Now) — элемент управления ActiveX, который позволяет покупателю приобретать товары с любой Web-страницы в Internet. Когда покупатель щелкает мышью на изображении продукта, появляется диалоговое окно для ввода адреса доставки (отгрузки) и данных о кредитной карте.

Cascading Style Sheet (каскадная таблица стилей) — описание форматирования, обеспечивающее расширенное управление внешним представлением и расположением HTML- и XML-элементов.

Catalog (каталог) — собирательный термин, служащий для обозначения индекса и хранилища свойств, в которых предусмотрен поиск. Каталоги размещаются на поисковом сервере хост-компьютера.

Catalog definition (определение каталога) — инструкции и параметры, заданные пользователем для построения каталога. Определения каталогов хранятся на сервере сборки каталога (catalog build server).

Catalog build server (сервер сборки каталога) — сервер, который производит сбор информации и осуществляет сборку каталогов.

Catalog schema (схема каталога) — описание столбцов или свойств и их атрибутов в каталоге Search.

CDATA section (CDATA-раздел) — раздел, который может быть применим для разметки тегов и зарезервированных символов, содержащих кавычки. Данные, размещенные в этом разделе, не интерпретируются.[1]

CDF file (CDF-файл) — см. *Channel Definition Format.*

CGI — см. *Common Gateway Interface.*

Channel Definition Format (формат описания канала) — формат данных на основе XML, используемый в Microsoft Internet Explorer для описания содержимого компонентов Active Channel и Active Desktop components.[2]

Clear Text/Basic Authentication (базовая аутентификация/аутентификация простым текстом) — метод аутентификации, в котором имя пользователя и пароль передаются в простом текстовом формате. Соединения должны шифроваться в соответствии с протоколом SSL.

Client Certificate Authentication (аутентификация по сертификату клиента) — метод аутентификации, в котором идентичность пользователя проверяет-

---

[1] Раздел CDATA — часть XML-документа, в которой разметка (кроме той, что указывает конец раздела CDATA) не интерпретируется, а передается в приложение в неизменном виде. — *Пер.*

[2] CDF — словарь XML, созданный Microsoft, позволяющий разработчику использовать различные механизмы доставки для опубликования коллекций информации, именуемых каналами, от Web-сервера к любому устройству, работающему в Internet. — *Пер.*

ся без использования пароля. Проверка выполняется путем вычислений, в которых используется открытый ключ (public key), хранящийся в цифровом сертификате пользователя, и закрытый ключ, находящийся в компьютере пользователя.

Common Gateway Interface (CGI, стандартный интерфейс шлюза, интерфейс общего шлюза) — интерфейс для запуска внешних программ или шлюзов под управлением сервера информации (information server). Шлюзы представляют собой программы, которые быстро обрабатывают информационные запросы и возвращают соответствующие документы или же их генерируют.

Component (компонент) — объект, который инкапсулирует данные и код и обеспечивает строго определенный набор общедоступных служб (сервисов).

Component user (пользователь компонента) — пользователь компонентов Microsoft Site Server 3.0. Лицо, которое манипулирует компонентом после установки, выполненной администратором.

Container (контейнер) — объект, расположенный в дереве информационного каталога (DIT, Directory Information Tree), основным назначением которого является содержание других объектов.

content[1] **(содержимое, содержание)**

1. Любое количество информации или указанных материалов, содержащихся на сайте.
2. Коллекция файлов на хост-компьютере, предназначенная для доставки пользователям посредством Active Channels или Active Channel Multicaster.

cookie

1. Постоянный идентификационный код, назначенный пользователю, который позволяет отслеживать пользователя во время его посещений Web-сайта.
2. Файл, хранящийся на компьютере,обеспечивающий средства хранения и выборки информации на стороне клиента в соединении клиент-сервер, и предназначенный для использования сервером. Применяется для избежания дублирования ввода идентификатора пользователя (покупателя) при каждом повторном соединении.
3. Информация о клиенте в Internet, сохраняющаяся сервером Internet на компьютере клиента. Автоматически передается обратно с клиента на сервер, когда первый просматривает сайт.

---

[1] Content — это информация (в XML), расположенная между открывающим и закрывающим тегами. — *Пер.*

Cookie Identification (Cookie-идентификация) — метод идентификации, реализуемый под управлением Microsoft Windows NT Authentication, в котором "личность" пользователя выясняется при помощи глобально-уникального идентификатора (GUID, globally unique identifier) вместо имени пользователя. GUID хранится в cookie, ассоциированном с браузером пользователя.

Crawl (последовательный обход)

1. Процесс, с помощью которого анализатор содержания (Content Analyzer), проверяет указанный вами сайт и создает карту. См. также *explore.*
2. Процесс, с помощью которого программа Search собирает содержание, последовательно обходя ссылки, находящиеся в документе, или обходя деревья каталогов в файловой системе.

crawl catalog (crawl-каталог) — каталог, составленный путем последовательного обхода документов в Internet, intranet и файловых системах.

Credentials (верительные данные) — имя и пароль пользователя, глобально-уникальный идентификатор (GUID, Globally unique identifier), или сертификат клиента, на основании которых проверяется подлинность пользователя.

Directory Information Tree (DIT, дерево информационного каталога) — представление в виде дерева всех вхождений в базе информационного каталога. DIT содержит узлы (nodes) — контейнеры и листья.

Distributed Component Object Model (DCOM, модель распределенных компонентных объектов) — протокол, который позволяет компонентам программного обеспечения надежно, безопасно и эффективно взаимодействовать в сети. DCOM разработан для использования в сетях с разными транспортными протоколами, включая протоколы Internet, например, HTTP.

Globally unique identifier (глобально-уникальный идентификатор) — строка, обычно числовая, которая уникальным образом идентифицирует элемент, операцию или отдельного человека. Идентификаторы shopperID и orderID, используемые в Commerce Server, являются глобально-уникальными идентификаторами.

GUID — см. *Globally unique identifier.*

Host (хост, хост-компьютер) — компьютер, имеющий IP-адрес.

Inline link (внутренняя ссылка) — ссылка, указывающая на ресурс в данной странице.

LDAP — см. *Lightweight Directory Access Protocol.*

Lightweight Directory Access Protocol (LDAP, облегченный протокол доступа к сетевому протоколу) — протокол доступа в Internet, позволяющий клиентским Internet-приложениям получать доступ к службам каталога. P&M использует LDAP для обеспечения доступа клиента к каталогу принадлежности (Membership Directory).

Membership Directory (каталог принадлежности) — центральный репозиторий данных о пользователях для Microsoft Site Server version 3.0, который содержит наборы параметров пользователя и информацию адресной книги.

Metadata (метаданные) — информация о данных.

Microsoft Management Console (MMC, консоль управления, административная консоль) — интегрированный пользовательский интерфейс для администрирования, который может одновременно использоваться различными приложениями, посредством модулей администрирования (snap-ins, administration modules), и который заменит IIS Internet Service Manager в IIS 4.0, а также различные средства администрирования Microsoft Windows NT Server.

Open Database Connectivity (ODBC, открытые средства связи с базами данных) — технология, которая обеспечивает общий интерфейс для доступа к гетерогенным (разнородным) базам данных SQL.

OPP — см. *Order processing pipeline.*

Order ID (идентификационный номер заказа) — строка, хранящаяся в URL HTML-страницы Commerce Server. Этот номер назначается с началом действий покупателя по оформлению заказа и используется для отслеживания заказа в течение его выполнения.

Order processing pipeline (OPP, конвейер обработки заказа) — серия выполняемых действий или ступеней, которые могут потребоваться после того, как покупатель разместит заказ. Например, в число ступеней обработки может входить проверка заказа, вычисление налогов и стоимости отгрузки.

P&M Service Administration — инструментальное средство P&M, управляющее распределением памяти сервера приложения и его безопасностью службами P&M: AUO, P&M Authentication Service и LDAP Service.

pcf file (pcf-файл) — файл, содержащий данные о выполнении заказа и конфигурации конвейера. Иногда называют файлом конфигурации конвейера.

Public area (общая область) — множество Web-страниц или дерево каталога, содержание которых доступно для любого пользователя без каких-либо ограничений (при отсутствии идентификации, аутентификации или разрешения).

Receipt (квитанция подтверждения) — подтверждение заказа на закупку в HTML-коде, выработанное Commerce Server в ответ на успешное выполнение транзакции закупки. Квитанция подтверждения содержит учетный номер заказа (order-tracking number), который может использоваться покупателем для напоминания об обязательствах по его выполнению. Если конвейер и база данных Commerce Server сконфигурированы для возможности сохранения квитанций, то история выполнения заказа может отображаться по требованию пользователя или администратора сайта.

Secure Sockets Layer (SSL, протокол безопасных соединений) — протокол соединения, поддерживающий аутентификацию сервера и (необязательно) клиента, на основе сертификатов по открытому ключу. Он также обеспечивает взаимодействие служб шифрования и/или целостности при их использовании в передаче данных во время сеанса.

ShopperID (идентификатор покупателя) — строка из 32-х символов, сгенерированных случайным образом сервером коммерции для отслеживания заказа покупателя. Идентификатор покупателя формируется, когда покупатель впервые входит на сайт.

Site hop — "прыжок", совершаемый из документа на одном сайте в документ на другом сайте.

Snap-in — модуль программного обеспечения, который интегрируется с MMC и используется для создания инструментальных средств администрирования.

SSL — см. *Secure Sockets Layer.*

Text markup (разметка текста) — включение тегов в текстовый поток элемента для разметки определенных частей элемента с помощью дополнительной метаинформации.

Wallet — управляющий элемент ActiveX, который позволяет пользователю вводить информацию о платеже и адресе при инициировании закупки в режиме online. Элементы управления — **Payment Selector** и **Address Selector** — размещены на компьютере покупателя и обеспечивают выпадающие меню, содержащие информацию о кредитных картах и адресах, хранимых в текущий момент времени. Покупатели могут вводить новые данные или выбирать их из уже существующих.

Windows Scripting Host (WSH) — независимый от языка хост для скриптов на основе Microsoft Windows NT Server 5.0 для 32-битной платформы Windows. Включает механизмы VBScript и JavaScript.

World Wide Web Consortium (WC3) — международный консорциум, основанный в 1994 г. для разработки стандартов для Web.

WSH — *см. Windows Scripting Host.*

XML document (XML-документ) — объект данных, который в соответствии с рекомендациями XML, является правильным (well-formed) и может (но не обязан) быть допустимым (действительным) (valid).

XML Engine (механизм XML) — программное обеспечение поддержки функциональности XML на стороне клиента.

XML parser (синтаксический анализатор XML) — XML-процессор, используемый для чтения XML-документов и обеспечивающий доступ к их содержанию и структуре.

XSL Formatting Objects (объекты форматирования XSL) — набор семантик форматирования, выраженный в виде словаря XML. Часть XSL.

XSL Patterns (шаблоны XSL) — часть XSL, которая обеспечивает простые возможности по запросам относительно XML-документа.

XUL — это XML для пользовательских интерфейсов.

Предметный указатель

O

P

S

T

W

X

Т

Ц

Ш

Э

Я