ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ РФ
ДИМИТРОВГРАДСКИЙ ИНСТИТУТ ТЕХНОЛОГИИ, УПРАВЛЕНИЯ И ДИЗАЙНА
УЛЬЯНОВСКОГО ГОСУДАРСТВЕННОГО ТЕХНИЧЕСКОГО УНИВЕРСИТЕТА
Кафедра «Математики и информационных технологий»

Курсовая работа
по курсу: “Технология разработки программного обеспечения”

Тема: Разработка Интернет-магазина на языке Java

Руководитель: Нестеркин В.Н.

_____
(подпись) Ф.И.О.
_____(дата)

Студент:

Потеренко А.Г.

Группа ВТ-41

_____(дата)

654600 (220400)
шифр направления (код специальности)

Проект защищен
_____
(дата)
с оценкой_____

Димитровград
2007

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данная курсовая работа представляет собой взаимосвязь таких важных понятий как электронная коммерция и электронный бизнес. Процесс их объедения с современными технологиями является главной задачей, поставленной перед автором.

Электронная коммерции – технология, позволяющая заказчикам оформлять и оплачивать заказы на веб-узле предприятия. Предприятия используют свой веб-узел не только как постоянно обновляемый рекламный проспект, но дают возможность заказчикам совершать сделки на этом веб-узле и могут даже дополнительно предоставлять услуги и поддержку в оперативном режиме. Для этого обычно применяются те или иные формы защищенных транзакций, основанных на использовании технологий, которые будут. Это позволяет предприятию проводить свои деловые операции круглый год, 24 часа в сутки, что может привести к расширению торгового оборота, уменьшить затраты на сбыт и обслуживание клиентов, а также повысить качество обслуживания заказчиков.

Электронный бизнес – полная интеграция технологии Internet в экономическую инфраструктуру предприятия. Предприятия применяют технологию Internet во многих областях своей деятельности. Для управления всеми внутренними и внешними процессами применяются сети, а сбыт, обслуживание и реклама осуществляются исключительно через веб. Кроме прочих потенциальных преимуществ, на предприятии ускоряется обмен данными, деловые процессы становятся проще, эффективнее, а производительность возрастает.

Умение управлять процессом внедрения современные технологий в инфраструктуру предприятия является главной задачей программистов, работающих на данном предприятии. Перед ними стоит очень сложная задача, решить которую они обязаны качественно и в относительно короткий срок.

Создать Интернет-магазин для большого предприятия требует очень больших капиталовложений. Современное программное обеспечение стоит очень дорого, поэтому задача программиста – найти золотую середину между выбором используемого программного обеспечения в процессе разработки проекта и созданием успешного проекта в кратчайшие сроки. Автор постарался выбрать именно те программные продукты, эффективность которых доказана уже не одним успешным проектом в мире электронного бизнеса.

1. ОБЩИЕ ВОПРОСЫ, СВЯЗАННЫЕ С РАЗРАБАТЫВАЕМОЙ СИСТЕМОЙ

1.1. Развернутая постановка задачи

Главная задача, поставленная перед автором:

1. Создание информационной системы, реализующей логику Интернет-магазина (и далее ИМ).

2. Создание удобного интерфейса пользователя через браузер для управления всей системой ИМ в целом.

3. Выбор программных средств для реализации поставленной задачи.

4. Программная логика должна быть реализована для трех пользователей системы: клиента, рабочий магазина, администратора магазина.

5. Система должна быть многопользовательской, как того подразумевает название проекта.

Веб-доступ подразумевает использование веб-интерфейса для доступа к СУБД. В дальнейшем разработанные модули понадобятся предприятию для полного перевода всей интерфейсной логики на уровень веб.

У автора на выбор предоставляется много технологий программирования и систем управления базами данных. Поэтому второстепенная задача – провести анализ существующих средств и правильно сделать выбор.

1.2. Анализ аналогичных разработок

В Интернете можно найти много компаний, которые специализируются на создании разработок, призванные решить поставленную задачу:

1. Компания "WEB-студия BB2". Программисты этой фирмы давно себя зарекомендовали с хорошей стороны. Их разработки удовлетворяют современным требованиям построения интерфейса для пользователя. Но есть у них два больших недостатка. Первый – цена ($2500) за каждый заказ (рекомендуется примерно от 500 до 1000 $). Второй – они используют технологии создания приложений, о которых заказчик может только догадываться. На мой взгляд, заказчик должен быть уверен в безопасности, масштабируемости разрабатываемой системы и должен заранее знать о том, как конфиденциальные данные будут передаваться по сети (в шифрованном, нешифрованном виде), как, и где будут храниться данные (в какой базе), на какой технологии будет работать сервер приложений и т.п. Вся эта информация закрыта для заказчика.

2. "1С: Аркадия Интернет-магазин". Хотя фирма 1С зарекомендовала себя с лучшей стороны в области бухгалтерского учета и торговли, для крупномасштабных проектов она непригодна. Для выполнения длитель-

ных транзакций используется SQL-версия продукта 1С. Это уже большие накладные расходы на сервер MSSQL. Без него в сети с большим числом пользователей (каким является Интернет) использование данной разработки не имеет смысла. К тому же версия 1С не многоплатформенная, так как она базируется на ОС семейства Windows, как известно – самой небезопасной системы с большим числом ошибок, которые легко эксплуатируют хакеры с помощью эксплоитов.

Первый вариант подходит предприятиям, не имеющих собственного хостинга, соответственно собственной сети с сервером приложений и СУБД.

Второй накладывает ограничение на платформу и зависимость от небезызвестной фирмы Microsoft.

2. АНАЛИЗ СРЕДСТВ И МЕТОДОВ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

2.1. Выбор технологии программирования

Для Интернета сейчас существуют много языков программирования. В числе них наиболее известные C#, PHP, Java.

C# – творение Microsoft. Язык, наряду с технологией ASP .Net позволяет создавать код, генерирующий динамические страницы. Об этой платформе можно рассказывать бесконечно. На мой взгляд, самая удачная разработка этой корпорации. Microsoft .Net – это платформа, включающая серверы, клиенты и сервисы. Она содержит набор приложений, таких, как Visual Studio .Net, Tablet PC, и .Net My Services. Microsoft .Net была спроектирована таким образом, чтобы удовлетворять текущие и будущие требования заказчиков к созданию, внедрению и управлению приложениями. Фундаментальным принципом Microsoft .Net является изменение процесса разработки и внедрения программного обеспечения – в частности, переход от разработки собственных программ к покупке, настройке и интеграции готового программного обеспечения. Microsoft .Net была разработана для интеграции приложений с помощью Web-служб, используя протоколы и форматы, такие, как SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language) и UDDI (Universal Description, Discovery, and Integration). Многоязыковая поддержка также является большим плюсом для этой платформы.

Можно сказать только одно про эту платформу. Новизна этой технологии должна зарекомендовать себя временем. Продукт создан одной корпорацией и учитывает не все требования, которые выдвигают множества мелких компаний, поэтому не все стремятся переходить на .NET.

Язык PHP – сравнительно молодой, но в то же время удивительно удобный и гибкий язык для программирования веб. С помощью него можно написать 99% программ, которые обычно требуются в Интернете. Для оставшегося 1% придется использовать C или Perl. Главный недостаток на мой взгляд – язык очень расположен к ошибкам, допускаемых программистами низкого и среднего уровня. Если и разрабатывать на PHP, то необходимо знать в корне все его особенности, иначе приложение просто опасно использовать. Не счесть случаев использования глубоких знаний этого языка в корыстных целях, которые преследуют хакеры. Существуют много реальных примеров взлома Интернет-магазинов, построенных на технологии PHP. Автор не рискнул его использовать в качестве технологии для своего проекта.

Язык Java – творение Sun Microsystems. Отличительная особенность Java - средства безопасности. Как язык, так и сама платформа с самого начала разрабатывались с учетом безопасности. Java-платформа позволяет пользователям загружать по сети непроверенный код и запускать его в безопасной среде, в которой он не сможет нанести какой-либо ущерб. Непроверенный код не сможет заразить базовую систему каким-либо вирусом, не сможет считывать или записывать файлы на жесткий диск и т. п. Одни эти возможности делают Java уникальной платформой.

В Java любой Java-код – апплет, сервлет, JavaBeans-компонент или все Java-приложение – можно запускать с ограниченными правами доступа, что предотвратит возможность нанесения какого-либо ущерба системе.

Вопросы безопасности, связанные с языком и платформой Java, тщательно рассматривались специалистами во всем мире. На ранних этапах существования Java ошибки в системе безопасности - некоторые из них были достаточно серьезные - не раз обнаруживались, но в дальнейшем были исправлены. Java предоставляет надежные гарантии, связанные с безопасностью, и поэтому случаи, когда находится новая ошибка в системе безопасности, становятся сенсацией. Ни одна другая распространенная платформа не предоставляет таких серьезных гарантий безопасности, как Java. Никто не утверждает, что в будущем не будут обнаружены новые прорехи в системе безопасности Java, но даже если безопасность Java не является совершенной, эта платформа на практике показала себя достаточно надежной в повседневном использовании. Безусловно, она лучше всех других альтернатив.

Технология Java поддерживается и совершенствуется с помощью так называемого процесса Java Community Process (JCP), представляющего собой взаимодействие более 400 компаний, организаций и частных лиц в целях создания платформы для сервисов и приложений, которая может работать на системах любого типа.

Java приложения могут выполняться на любых операционных системах: системах уровня предприятия, таких, как Unix, Linux, OS/390, Windows 2000, или HP-UX; домашних операционных системах, таких как Mac OS, Windows или Linux; а также операционных системах для мобильных устройств, таких как Palm OS или Symbian EPOC

Можно также отметить, что технология Java является открытой и построена на внутриотраслевых стандартах для программного обеспечения. Любой желающий может загрузить и изучать код Java платформы.

Итак, основываясь на вышеизложенном высказывании, можно сказать, что разработка проекта именно на Java имеет много перспектив в будущем.

2.2. Технологии от Sun

При обсуждении Java важно понимать различия между языком программирования Java, виртуальной машиной Java и Java-платформой. Язык программирования Java - это язык, на котором написаны Java-приложения (включая апплеты, сервлеты и JavaBeans-компоненты). Когда компилируется Java-программа, она переводится в байт-коды, представляющие собой переносимый машинный язык архитектуры центрального процессора, который известен как виртуальная машина Java (другие названия: Java VM и JVM). JVM может быть реализована напрямую в аппаратном обеспечении, но обычно она реализуется в форме программного обеспечения, которое интерпретирует и выполняет байт-коды.

Java-платформа отличается как от языка Java, так и от Java VM. Java-платформа является предопределенным набором Java-классов, которые существуют в каждой Java-инсталляции; эти классы могут использоваться всеми Java-программами. Еще Java-платформу иногда называют средой выполнения Java или основными Java APIs (прикладными программными интерфейсами). Java-платформа может быть расширена при помощи дополнительных стандартных расширений. Такие API-расширения существуют в некоторых Java-инсталляциях, но нет никакой гарантии, что они присутствуют в каждой из них.

В нашей работе нас, прежде всего, интересуют две платформы Java 2 Platform Standard Edition и Java 2 Platform Enterprise Edition. Первая предоставляет интерфейс для прикладных программ, например апплетов, сетевых приложений. Вторая позволяет создавать приложения уровня предприятия.

Используя API первой платформы, мы напишем апплет и создадим некоторые классы для шифрования и сетевого взаимодействия.

Из второй платформы нам понадобятся две технологии: JSP и сервлеты. Именно на них и базируется данная разработка.

2.2.1. Технология сервлетов

Сервлет Java – это серверная программа, вызываемая web-сервером для обслуживания HTTP-запросов. Он работает в виртуальной машине Java (Java Virtual Machine) на веб-сервере и обычно выполняет какие-то вычисления, генерируя содержимое HTTP-ответа. По сравнению с CGI и другими методами серверного программирования сервлеты лучше, потому что:

- в отличие от программы CGI, при каждом вызове сервлета не создается новый процесс. Сервлеты вызываются, как правило, посредством

потоков, обслуживающих отдельные HTTP-запросы. Число одновременно порождаемых потоков можно ограничить, что помогает избежать перегрузки сервера;

- это обычные программы Java, транслируемые в не зависящий от платформы байтовый код, и поэтому их можно автоматически переносить на любую машину, где применяется Java. Работают они в области, защищенной границами виртуальной машины Java, и не могут быть причиной нарушения правил доступа к памяти и вызывать аварии сервера;
- сервлеты имеют полный доступ к мощным прикладным программным интерфейсам Java, таким как Enterprise Java Beans, Java Mail, JNDI и RMI.

2.2.2. Технология JSP

Технология JavaServer Pages (JSP) встроена в описанную выше модель сервлетов и основана на смешивании программных конструкций Java со статичными шаблонами HTML или XML. Логические схемы Java призваны генерировать динамическое содержимое страницы, а язык разметки – структурировать и представлять данные. Главная цель проекта JSP – строго отделить текст HTML или XML от программ Java специально для web-дизайнеров, которые незнакомы с программированием на Java. С такой целью в спецификации JSP предусмотрены встроенные средства компонентного программирования при помощи JavaBeans и библиотек тегов.

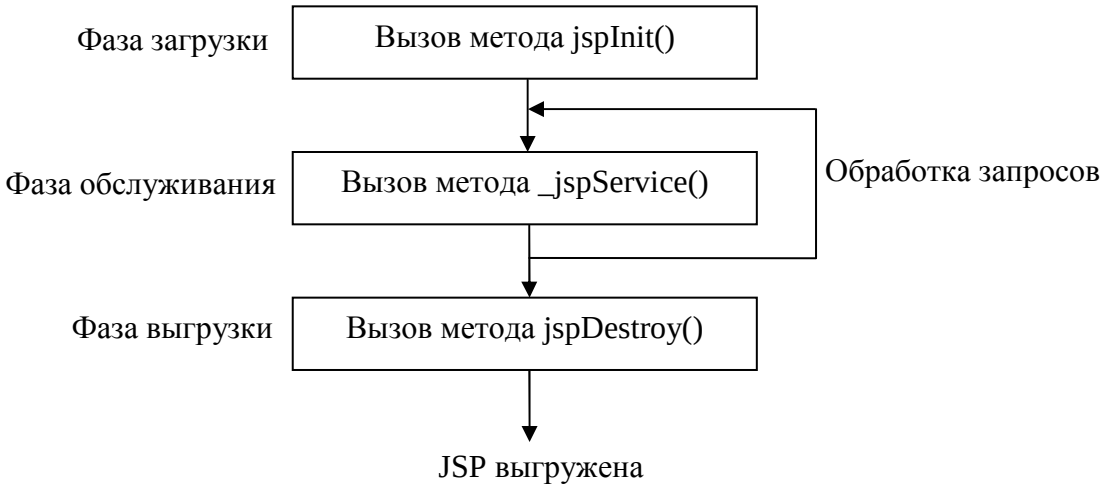


Рис. 1. Модель выполнения JSP.

Все JSP-программы выполняются двигателем JSP, работающим в границах веб-сервера. Обычно двигатель JSP функционирует в качестве сервлета, вызываемого тогда, когда URL запроса заканчивается расширением *.jsp. Такая схема вызова называется *отображением расширений* (extension mapping).

После вызова двигатель JSP сначала находит JSP-файл, на который ссылается URL. Если файл не был преобразован заранее, двигатель JSP превращает "смешанный" исходный текст JSP в код сервлетов Java. Генерируемый код соответствует правилам, установленным спецификацией JSP, и в нем используются классы JSP этапа выполнения. Генерируемый класс сервлета реализует стандартный интерфейс javax.servlet.jsp.HttpJspPage. Этим интерфейсом расширяется супертип javax.servlet.jsp.JspPage, имеющий два метода: jspInit() и jspDestroy(). Они вызываются двигателем JSP для инициализации и уничтожения генерируемого сервлета. В интерфейсе javax.servlet.jsp.HttpJspPage содержится еще один метод – _jspService(), генерирующий выходную страницу и такой же полезный в многопоточной обработке HTTP-запросов, как метод service() в сервлетах. В _jspService() содержатся динамические конструкции, например JSP-выражения и скриптлеты, встроенные в страницу JSP. Генерируемый сервлет компилируется в байтовый код Java с помощью стандартного компилятора Java. Наконец, сгенерированный класс сервлета загружается и выполняется двигателем сервлетов, а результаты возвращаются клиенту.

Последующие или одновременно поступающие запросы обращаются к тому же методу JspService(), используя новый поток, порождаемый двигателем сервлетов. Так что нужно следить за безопасностью потоков в программном тексте JSP. После окончательной выгрузки JSP-класса (например, при остановке web-сервера) двигателем JSP вызывается метод jspDestroy(). Очевидно, что жизненный цикл JSP очень похож на жизненный цикл сервлета.

2.2.3. Технология апплетов

Одна из целей разработки Java – это создание апплетов, которые являются маленькими программами, запускаемыми внутри браузера. Поскольку они должны быть безопасны, апплеты ограничены в своих возможностях. Однако апплеты являются мощным инструментом для поддержки программирования на стороне клиента – главной способности для веб.

Программирование апплетов настолько ограничено, что часто рассматривается как пребывание “внутри песочницы”.

Апплет не может касаться локального диска, то есть апплет не может прочитать информацию с машины клиента и передать ее злоумышленнику.

Но Java предлагает цифровую подпись для апплетов. Многие ограничения апплетов освобождаются, когда вы согласитесь доверить апплету

(который подписан источником, которому вы доверяете) доступ к вашей машине.

Апплеты имеют определенные преимущества, особенно при построении клиент/серверных или сетевых приложений:

- не требуется установки. Апплет имеет независимость от платформы;
- не нужно беспокоится о плохом коде, являющемся причиной крушения чьей-то системы, потому что система безопасности встроена в ядро языка Java и в структуру апплета.

Наиболее корректным будет следующее определение апплета – это программа, созданная на основе класса java.applet.Applet или на основе одного из подклассов этого класса.

2.3. Выбор СУБД

Построение современных информационных систем сегодня напрямую связано с реляционными и объектно-ориентированными СУБД. Данные системы в последнее время утвердились как основные средства для обработки данных в информационных системах различного масштаба от больших приложений обработки транзакций в банковских системах до персональных систем на PC. В настоящее время существует множество систем управления базами данных (СУБД) и других программ выполняющих сходные функции. Инструментальные средства Oracle Database 9i – одни из лучших и наиболее мощных имеющихся инструментов разработки профессионального класса.

В современном мире существует как минимум три направления для выбора СУБД для своих проектов. Первое направление – Microsoft со своим MSSQL Server. Программисты этой СУБД неплохо выполнили свою задачу. Главный недостаток – данная СУБД работает только на платформе Windows. Второе направление – OpenSource СУБД, такие как MySQL и Firebird. Хотя первая и предназначена для работы в Web, но не имеет поддержку длительных транзакций. Вторая вовсе предназначена для малых систем управления. Третье направление – СУБД Oracle. Перенесена под все существующие платформы. Поддерживает сверхбольшие БД (свыше 1 ТБ). Мощная и удобная система управления транзакциями. Встроенный язык PL/SQL со своими модулями DBMS позволяет сделать почти все для надежной и эффективной работы. Поддержка хранимых процедур на языке Pro*C, Pro*C++, Java. Существуют множество драйверов под различные языки для взаимодействия с БД, таких как Perl, PHP, Java. Пакет Oracle Database 9i, наделенный самым развитым набором функций для работы с языком Java и доступа к данным через Интер-

нет, является системой оптимизации одновременного доступа. Единственным недостатком данной СУБД является сложность администрирования, однако все затраты на ее внедрение и освоение в последствии окупятся эффективной и надежной работой. По уровню производительности при работе в веб-среде под Linux Oracle занимает почетное второе место после OpenSource СУБД MySQL, при этом значительно превосходя все другие СУБД (в том числе и СУБД MySQL) по надежности и безопасности.

Для интеграции Oracle Database 9i и различных Интернет-технологий (в первую очередь всемирной паутины) и предназначена данная курсовая работа. При использовании данной разработки взаимодействие пользователей с базой данных Oracle Database 9i можно организовать, совместив средства этой СУБД и стандартные возможности веб-интерфейса. Веб-приложения на основе Oracle могут входить в структуру внутрикорпоративных процессов управления, так и представлять собой Интернет-магазины.

Данная разработка может использоваться в любом учреждении, предприятии или организации, которая использует систему Oracle Database 9i для автоматизации своей деятельности и испытывает необходимость в электронной коммерции.

2.4. Выбор способа доступа к БД

Интернет-магазин по своей сути предполагает активное взаимодействие с какой-либо базой данных для различных целей. Например, для хранения каталогов товаров, информации о зарегистрированных покупателях, их заказов и т.д.

Именно использование Oracle 9i Database Server, как СУБД, позволяет решить задачу организации электронной торговли, автоматизации работы предприятия.

Для доступа к данной СУБД создано множество драйверов для каждого из языков программирования. Мы же будем использовать механизм под названием JDBC от Sun, так как реализацию будем делать на Java.

Летом 1996 года компания Sun выпустила первую версию интерфейса для организации доступа Java-приложений к базам данных JDBC (Java Database Connectivity). Настоящий интерфейс позволяет программистам соединяться с базой данных, запрашивать и обновлять данные с помощью языка структурированных запросов SQL. Этот язык фактически стал стандартным средством взаимодействия с реляционными базами данных.

Java и JDBC имеют весомое преимущество по сравнению с другими инструментами для работы с базами данных. Программы, созданные с помощью Java и JDBC, не зависят от используемой платформы и поставщика программного обеспечения.

Одна и та же программа на языке Java может одинаково хорошо работать на настольной системе под управлением Windows NT, на сервере Solaris. При этом допускается перемещение данных из одной базы данных в другую, например из Microsoft SQL в Oracle. В результате чтение данных может выполнять одна и та программа. Именно эта возможность выгодно отличает Java и JDBC от других способов программирования баз данных.

В течение многих лет было разработано большое количество эффективных и надежных приемов работы с базами данных. Стандартные реляционные базы данных поддерживают индексы, триггеры, хранимые процедуры и инструменты управления транзакциями. JDBC предоставляет все эти возможности.

Суть технологии JDBC заключается в следующем: производитель СУБД создает свой драйвер для доступа к своей СУБД, а Java API предоставляет диспетчер драйверов. Благодаря этому к диспетчеру могут подключаться драйвера различных СУБД. Драйвера должны соответствовать API диспетчера драйверов.

В результате было создано два интерфейса. Разработчики приложений используют JDBC API, а поставщики баз данных и инструментальных средств – JDBC Driver API.

Подход, используемый при создании JDBC, основан на очень успешной модели ODBC-интерфейса компании Microsoft.

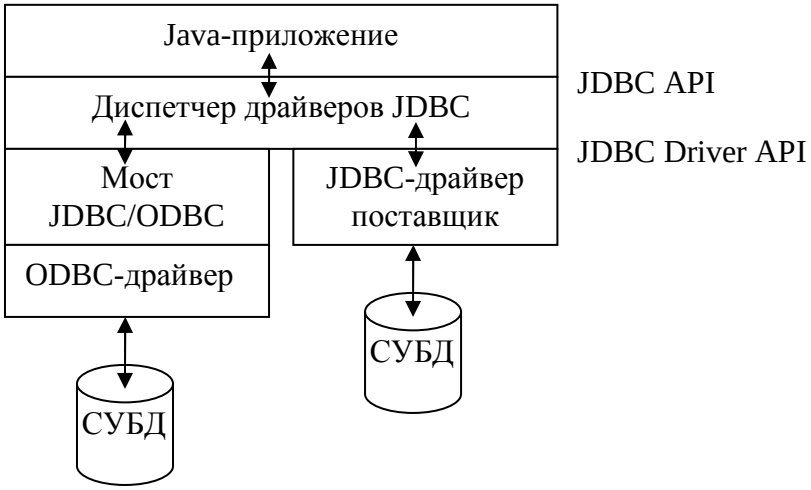


Рис. 2. Взаимодействие JDBC и базы данных

Каждый JDBC-драйвер принадлежит одному из перечисленных ниже типов.

- Драйвер типа 1. Транслирует JDBC в ODBC и для взаимодействия с базой данных использует драйвер ODBC. Компания Sun включила в состав JDK один такой драйвер – мост JDBC/ODBC. Однако для его использования требуется соответствующим образом установить и конфигурировать ODBC-драйвер.
- Драйвер типа 2. Создается преимущественно на языке Java и частично на собственном языке программирования, который используется для взаимодействия с клиентским API базы данных. Для использования такого драйвера нужно помимо библиотеки Java установить специфический для данной платформы код.
- Драйвер типа 3. Создается только на основе библиотеки Java, в которой используется независимый от базы данных протокол взаимодействия сервера и базы. Этот протокол позволяет транслировать запросы в соответствии со спецификой конкретной базы. Если код, зависящий от базы данных, находится только на сервере, доставка программ существенно упрощается.
- Драйвер типа 4. Представляет собой библиотеку Java, которая транслирует JDBC-запросы непосредственно в протокол конкретной базы данных.

В нашем случае необходимо рассмотреть драйвера непосредственно от компании Oracle. Oracle предоставляет следующие JDBC драйвера:

- Thin драйвер, 100% Java драйвер для клиентской стороны без Oracle инсталляции. Может использоваться с апплетами и приложениями;
- OCI драйвер для клиентской части с использованием Oracle инсталляции; может использоваться только в приложениях;
- server-side Thin driver, который по функциональности является таким же как обычный Thin драйвер, но весь код выполняется внутри Oracle Server;
- server-side internal driver, весь код выполняется внутри Oracle Server.

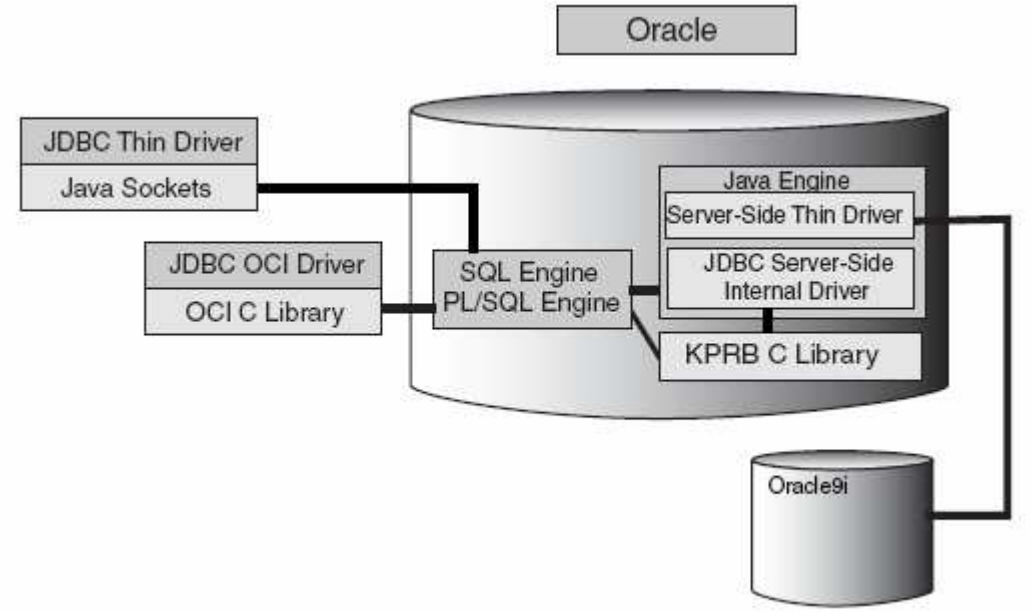


Рис. 3. JDBC-драйвера Oracle

Для нас вполне подойдет Thin драйвер, не требующий инсталляции. Данный драйвер представляет собой две библиотеки с расширением *.jar. Это classes111.jar и nls_charset11.jar. Пути к этим библиотекам должны быть объявлены в переменной окружения CLASSPATH, для того чтобы компилятор Java нашел их.

2.5. Выбор веб-сервера

Так как условиями задачи предопределено использование Java-технологии для решения задачи, значит, нам необходимо использовать тот сервер, который поддерживал бы элементы J2EE (напомним, что таковыми элементами являются JSP и сервлеты). На выбор нам предоставляется множество различных веб-серверов. В том числе и от фирмы Oracle, но, на мой взгляд, для нашей задачи подошли бы два из всего множества. Это веб-сервер Blazix и Jakarta Tomcat Servlet/JSP Container. Первый представляет собой быстродействующий сервер приложений Java. Предельно прост в настройке и администрировании. Бесплатен для некоммерческого использования. Второй имеет смысл использовать в том случае, когда будет необходимость использовать наш проект как коммерческий. Итак, выберем Blazix версии 1.2.5.

2.6. Выбор инструментов для разработки

Для разработки программного комплекса будут применяться следующие системы программирования:

• NetBeans IDE 5.0. Распространяется по лицензии Sun Public License. Эта среда предназначена для создания проектов Java. Шаблоны коды позволяют быстро начать программировать. Встроенный отладчик, удобный редактор кода с подсветкой синтаксиса, встроенная справочная система по API. Программа работает на платформах Windows и Linux. Среда является бесплатным продуктом для коммерческих проектов;

• Htmlpad Fisherman 1.9. Бесплатная среда для создания JSP и HTML страниц. Удобный генератор HTML кода позволяет быстро создать интерфейс;

• PL/SQL Developer 6.0.6.947. Первоклассный инструмент для создания кода PLSQL. Первоклассный отладчик кода, браузер всех системных объектов СУБД Oracle, генератор всех известных конструкций встроенного языка и т.п. Продукт распространяется на коммерческой основе.

2.7. Выбор архитектуры разрабатываемой системы

Рассмотрим различные архитектуры построения приложений с использованием технологий Java.

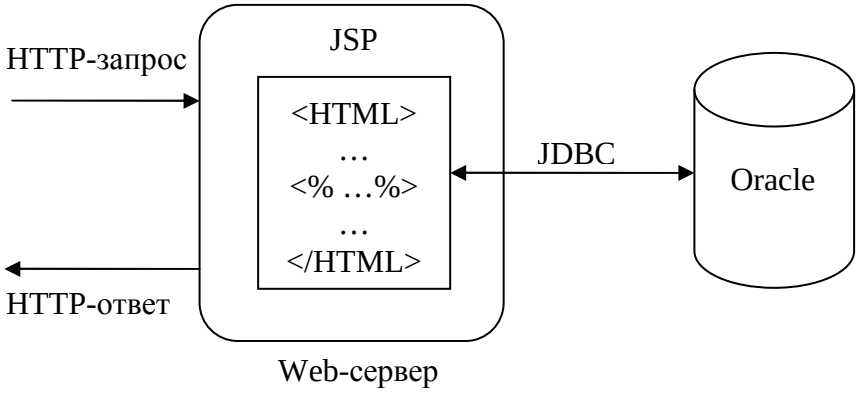


Рис. 4. Упрощенная архитектура приложения.

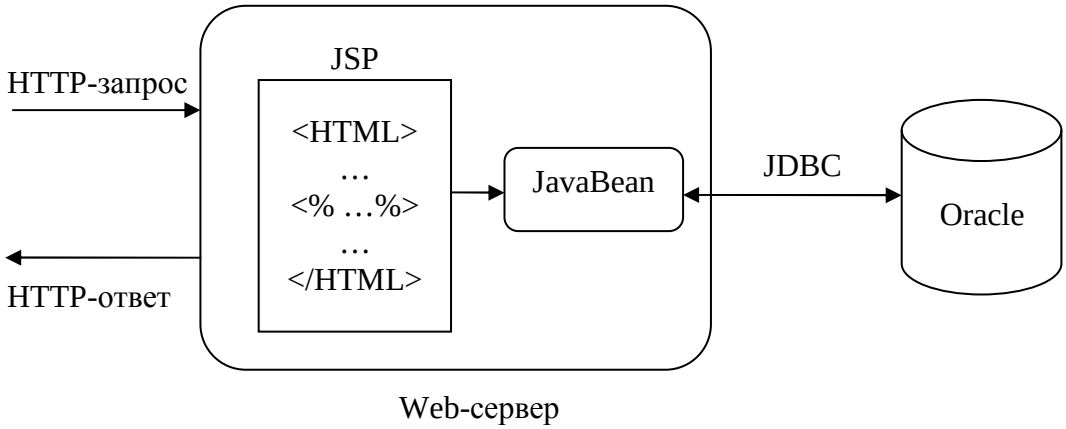


Рис. 5. Компонентная JSP-архитектура.

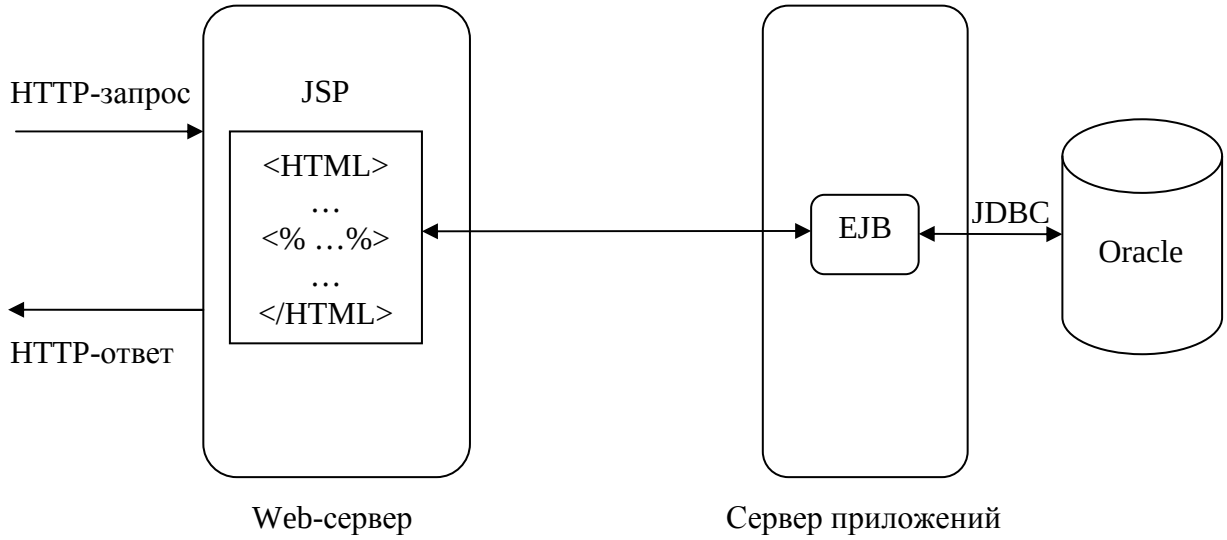


Рис. 6. Четырехуровневая архитектура.

Здесь представлены наиболее известные архитектуры. Теперь выберем ту, на которой будет базироваться наше приложение.

Первая архитектура крайне неудобная в программировании. Всю логику приходится выкладывать в JSP странице. В больших проектах страницы JSP будут загромождены тегами HTML и java-кодом. Приложение получится крайне не масштабируемым и в дальнейшем его будет трудно сопровождать.

Третья архитектура, которая характера для крупномасштабных распределенных проектов будет неуместна в нашей задаче. Сервер приложений с компонентой архитектурой зерен уровня предприятия Enterprise JavaBeans накладывает определенные ограничения. Вообще технология EJB не подходит для таких проектов как Интернет-магазин. В основном она используется, когда транзакций очень много в логике и вся работа с транзакциями происходит прозрачно для пользователя. Зайдем немного вперед и укажем, что в нашей задаче основная логика транзакций будет реализована в виде модулей PLSQL на сервере Oracle. Поэтому данная архитектура отпадает.

Осталась вторая архитектура. Удобная архитектура, которая позволяет создать абсолютно простой в сопровождении программный продукт. Всю основную нетранзакционную логику вынесем в компоненты JavaBean. Интерфейс пользователя выполним в виде JSP-страниц. Транзакции же будут реализованы в базе данных на языке PLSQL в виде хранимых объектов, таких как хранимые процедуры и функции.

2.8. Основные моменты при кодировании JavaBean, сервлетов и JSP

При написании JavaBean нужно обязательно следить за тем, чтобы программный код был безопасен с точки зрения потоков. В Java для надлежащей защиты программ от конфликтов при многопоточном выполнении следует использовать оператор synchronized. В частности, нужно синхронизировать доступ к переменным экземпляров класса JavaBean, например, к переменной, представляющей соединение с базой данных. Нередко необходимо группировать некоторые операции, выполняемые над базой, делая их частью атомарной транзакции, соответствующей одиночному HTTP-запросу. Таким образом, доступ к соединению с базой данных нужно координировать среди нескольких потоков, обслуживающих разные HTTP-запросы.

Запрет многопоточного доступа в некоторых случаях снижает производительность системы и эффективность обработки запросов. Некоторый код придется синхронизировать, что приводит к последовательной обработке одновременно поступающих запросов.

По своей природе HTTP – универсальный протокол, не сохраняющий состояние. Другими словами, HTTP-запрос никак не связан с любым другим, даже если оба поступают от одного и того же клиента. Однако при взаимодействии web-приложений состояние часто должно сохраняться. Известным примером взаимодействия с сохранением состояния являются сетевые магазины, когда пользователь собирает товары в виртуальную "корзину", заключающую в себе несколько запросов. Все это требует ввода понятия сеанса (session), в котором, по существу, группируются серии запросов от одного клиента.

Рассмотрим, как организуется работа с сеансами. Каждый сеанс идентифицируется на сервере уникальным ключом. Ключ сеанса может содержаться в сообщении HTTP-ответа, сохраняться клиентом и вводиться в последующие запросы, обращенные к данному серверу, указывая текущий сеанс. Распространены три способа контроля над сеансами:

- через домашние данные (cookies);
- путем перезаписи URL;
- посредством встроенного API для контроля над сеансами.

Заметим, что сеанс может быть завершен неявно по истечении времени сервером. По истечении срока действия сеанса его ключ становится недостоверным, даже если HTTP-запрос с данным ключом получен позже.

Домашние данные (cookies)

Это просто идентификационные данные сеанса, посылаемые клиенту в HTTP-заголовке Set-Cookie как часть сообщения HTTP-ответа. Программа может направить домашние данные явно, вызвав метод addCookie() объекта HttpServletResponse. Клиент вправе сохранить их и посылать во всех последующих запросах назад серверу для идентификации сеанса. Однако порой web-браузер не имеет поддержки домашних данных, либо пользователь явно запрещает их в браузере. В таких случаях применяется метод перезаписи (переадресации) URL.

Перезапись URL

При использовании этого метода все указатели URL, включаемые в ответ сервера, видоизменяются так, чтобы нести информацию о ключе сеанса. Например, идентификатор пользователя добавляется к URL в виде строки запроса:

http://localhost/GiperShop?userid=345HFGdT000FDJHDI767FDJBEU0

Когда впоследствии пользователь щелкает мышью на гиперссылке HTTP на web-странице, отображаемой браузером, для ссылки применяется этот расширенный URL. Другая страница извлекает идентификатор сеанса из строки запроса URL. Этот метод неудобен тем, что в ответе приходится модифицировать все указатели URL, вводя ключ сеанса, а это дополнительная нагрузка. Еще хуже то, что теряется конфиденциальность, поскольку идентификатор сеанса явно или неявно присутствует на генерируемой странице.

Сеансовый API

В прикладном программном интерфейсе (API) Java предусмотрена встроенная поддержка сеансов. С этим API должен работать любой web-браузер, поддерживающий J2EE. С помощью данного API программист, изолируется от всех деталей процесса слежения за сеансами, реализуемого web-сервером; отправка домашних данных или перезапись URL могут происходить в фоновом режиме.

Продемонстрируем, как должен использоваться этот метод на JSP-странице. Для этого нам необходимо рассмотреть тег <jsp: useBean>.

Этот тэг указывает на то, что определенное зерно используется на JSP-странице. Общий формат тэга:

<jsp: useBean id= "переменная" другие_атрибуты />

К другим атрибутам, помимо прочих, относятся class и scope. Наиболее часто используются атрибуты id, scope и class.

Атрибут class описывает полностью определенное имя класса для JavaBean.

Атрибут id описывает идентификатор Java для экземпляра зерна. С помощью этого идентификатора можно ссылаться на экземпляр зерна в той области действия Java, в которой присутствует тэг <jsp:useBean>.

Атрибут scope необязателен, но довольно важен. Он определяет время жизни (lifetime) компонента-зерна, используемого на JSP-странице, т.е. продолжительность его существования в качестве программной переменной.

Если атрибут scope будет равен session, то в этом случае экземпляр зерна связывается с объектом session и существует во время HTTP-сеанса, в котором вызывается данная JSP. При исполнении тэга <jsp: useBean> в объекте session создается экземпляр зерна, если, конечно, его там еще нет. Идентификатор зерна с областью действия session должен быть уникален среди всех страниц, вызываемых в границах одного и того же HTTP-сеанса.

Итак, мы определились, что будем использовать третий вариант для работы с сессией пользователя.

Каждая JSP-страница для клиента будет иметь вид:

<jsp:useBean id="BEAN" class="JavaBean_клиент " scope="session"/>

Каждая JSP-страница для персонала будет иметь вид:

<jsp:useBean id="BEAN" class="JavaBean_персонал" scope="session"/>

3. РЕАЛИЗАЦИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

3.1. Назначение системы

Система предназначена для обработки заказов от пользователей в режиме реального времени.

3.2. Описание системы

Система логически состоит из трех главных составляющих: раздел клиента (клиенты магазина), раздел для персонала магазина (главные пользователи) и административная часть (администраторы). Первые две из них должны быть реализованы через веб-интерфейс, последняя – в виде консольного приложения.

Система использует дополнительное программное обеспечение: веб-сервер Blazix, СУБД Oracle 9i для платформы Windows, Платформу J2EE и J2SE, FTP-сервер для каждого рабочего из персонала магазина Baby FTP Server. Последний необходим для клиента, написанного автором на Java для закачки изображений с пользовательской машины в базу данных.

Система физически написана с использованием четырех языков: Java, PL/SQL, HTML и Java-специфичный язык JSP.

Для своей работы требует три библиотеки, написанные на Java: classes111.jar, nls_charset11.jar, servlet-api-2.4.jar.

3.3. Технические характеристики

Ниже перечислены основные технические характеристики системы.

- Обработка заказов в режиме реального времени.
- Многопоточная обработка пользовательских запросов.
- Высокий уровень безопасности.
- Независимость клиентской части от операционной системы.
- Независимость пользовательской части от операционной системы.
- Независимость административной части от операционной системы.
- Управление данными с помощью СУБД Oracle 9i.
- Программное обеспечение веб-сервера – Blazix и Java-платформы J2EE и J2SE.
- Аппаратная платформа – Intel.

3.4. Административная часть

Для администратора было создано программное обеспечение, выполняющее следующие функции:

- регистрировать учетные записи персонала магазина. Администратор может ввести логин и пароль пользователя, программа добавит их в базу данных;
- удаление учетных записей персонала магазина. Удаление происходит по введенному логину;
- создание секретного симметричного ключа шифрования. Этот ключ в дальнейшем будет нужен для шифрования пароля;
- просмотр всех учетных записей персонала. Выводятся сведения об идентификаторе пользователя, его логине, пароле и времени последнего входа.
- дополнительно реализована функция для перевода jsp файлов из кодировки 1251 в кодировку ISO-8859-1.

Для запуска исполняемого модуля с помощью java интерпретатора необходимо передать ему параметры:

<SID> <host> <port> <key>

где SID – идентификатор экземпляра базы данных Oracle; host – машина, на которой работает сервер Oracle; port – удаленный порт для соединения; key – секретный файл с ключом.

Если необходимо воспользоваться дополнительной функцией, то необходимо передать главный параметр -ISO и три для обрабатываемых файлов jsp: папка, где содержатся обрабатываемые файлы, папка, куда будут созданы новые файлы и собственно обрабатываемый файл.

3.5. Пользовательская часть

Персонал магазина может входить в защищенную зону при помощи логина и пароля. Автором реализованы следующие функции управления:

- регистрация склада. При этом пользователь должен ввести данные о складе: название, страна, область, город, улица, дом;
- накладные прихода товаров. При этом пользователь должен выбрать ID склада и ID товара, ввести данные о прибывшем количестве товара;
- статистика по товарам и складам. Здесь пользователь может просмотреть данные о: суммарном количестве каждого товара по складам, суммарном количестве товаров на каждом складе, пустующих складах, закончившихся товаров;
- регистрировать новую категорию товаров. При этом пользователь должен ввести лишь название категории;

• регистрировать нового поставщика товаров. При этом пользователь должен ввести название поставщика;

• регистрировать новый способ оплаты заказов. При этом пользователь должен ввести название способа;

• регистрировать новый способ доставки заказов. При этом пользователь должен ввести название способа;

• регистрировать новый товар на складе. При этом пользователь должен ввести название товара, выбрать идентификатор категории товара, заполнить характеристики товара, выбрать бренд товара, заполнить описание товара, ввести гарантию на товар, выбрать идентификатор поставщика товаров, ввести цену товара, выбрать локальный файл с изображением товара;

• редактировать свойства товара. При этом пользователь должен выбрать идентификатор товара и продолжить редактирование;

• просмотреть товары по определенным критериям. Критерии включают в себя: идентификатор категории, идентификатор поставщика, бренд, гарантийный период (>, ≥, <, ≤, =), цена (>, ≥, <, ≤, =);

• просмотреть свойства отдельного товара;

• обрабатывать заказы пользователей.

В процессе решения поставленной задачи был реализован интерфейс, несколько отличающийся от интерфейса клиента. По функциональности он гораздо шире и сложнее. Его структура изображена на рисунке 7:

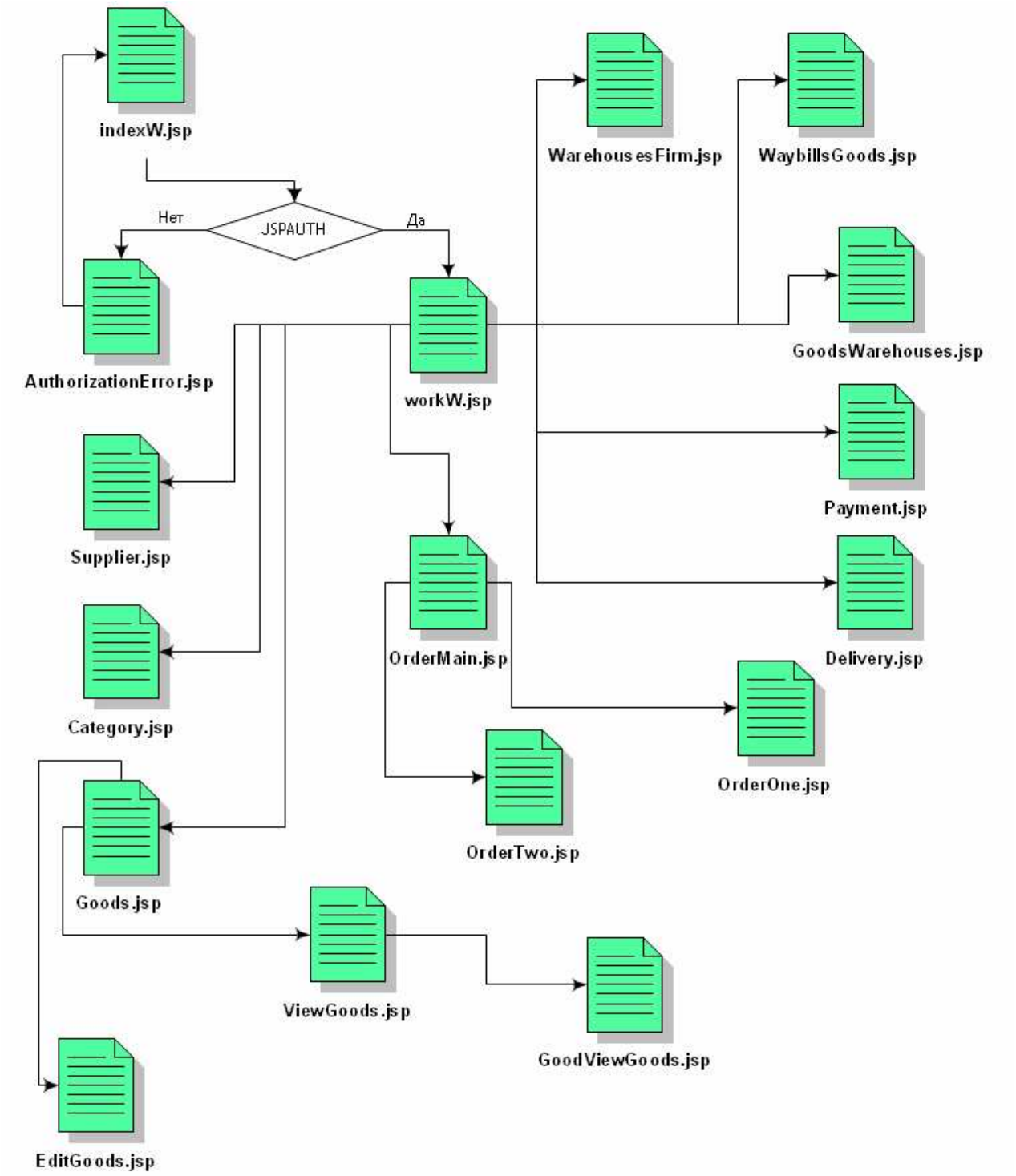


Рис. 7. Раздел сайта для персонала магазина.

Рассмотрим каждую страницу в отдельности.

• indexW.jsp – главная страница, с которой пользователь может авторизоваться;

• AuthorizationError.jsp – страница, на которую попадает пользователь в случае неуспешной авторизации;

• workW.jsp – страница, с которой пользователь начинает свою работу;

• WarehousesFirm.jsp – страница, на которой пользователь может работать со складами фирмы;

• WaybillsGoods.jsp – страница, на которой пользователь может работать с накладными прихода товаров;

• GoodsWarehouses.jsp – страница, на которой пользователь может просмотреть всю статистику по товарам и складам;

• Payment.jsp – страница, на которой пользователь может работать со способами оплаты заказов;

• Delivery.jsp – страница, на которой пользователь может работать со способами доставки заказов;

• Supplier.jsp – страница, на которой пользователь может работать с поставщиками товаров;

• Category.jsp – страница, на которой пользователь может работать с категориями заказов;

• Goods.jsp – страница, на которой пользователь может работать с товарами, то есть добавлять товары в базу данных;

• EditGoods.jsp – страница, на которой пользователь может редактировать свойства товаров;

• ViewGoods.jsp – страница, на которой пользователь может вести поиск товаров по определенным критериям;

• GoodViewGoods.jsp – страница, на которой пользователь просмотреть свойства отдельного товара;

• OrderMain.jsp – страница, на которой пользователь может выбирать накопившиеся заказы на обработку;

• OrderOne.jsp – страница, на которой пользователь может просмотреть свойства заказа и выполнить его, укомплектовав нужные товары со склада;

• OrderTwo.jsp – страница, на которой пользователь может завершить заказ, удостоверившись, что от клиента поступила оплата.

3.6. Клиентская часть

Клиент имеет возможность работать в неавторизованном и авторизованном режиме. В первом случае он может регистрироваться, искать товары по контекстному поиску, просматривать товары. Во втором случае он должен авторизоваться или зарегистрироваться. После этого ему будут доступны: добавление, удаление товаров из корзины, редактирование количества товаров в корзине. Переход на личную страницу, где ему будет выведена история его заказов, статус заказов, а также его личные данные.

Для пользователя был создан интерфейс, благодаря которому он может использовать все вышеизложенные функции. Его структура изображена на рисунке 8:

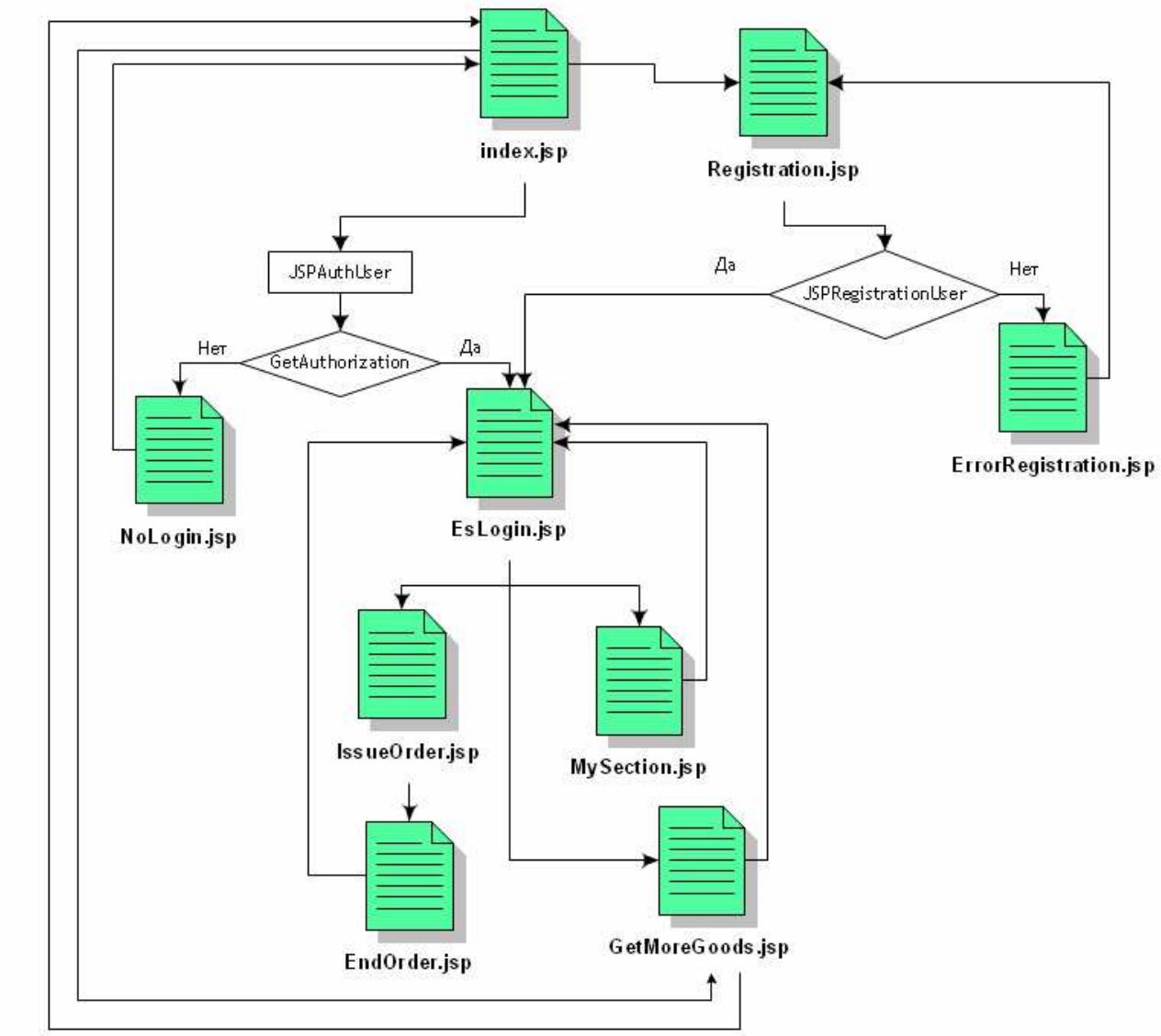


Рис. 8. Раздел сайта для клиента.

Рассмотрим каждую страницу в отдельности.

• index.jsp – главная страница, с которой пользователь начинает свою работу;

• Registration.jsp – страница, на которой пользователь может зарегистрироваться. Если регистрация проходит удачно, то клиент сразу же становится авторизованным пользователем и может совершать покупки, иначе ему повторно предоставляется возможность зарегистрироваться;

• EsLogin.jsp – страница, на которую переходит пользователь в случае успешной авторизации;

• NoLogin.jsp – страница, на которую переходит пользователь в случае неуспешной авторизации;

• ErrorRegistration.jsp – страница, на которую переходит пользователь в случае ошибки регистрации;

• IssueOrder.jsp – страница, на которой клиент может оформить заказ;

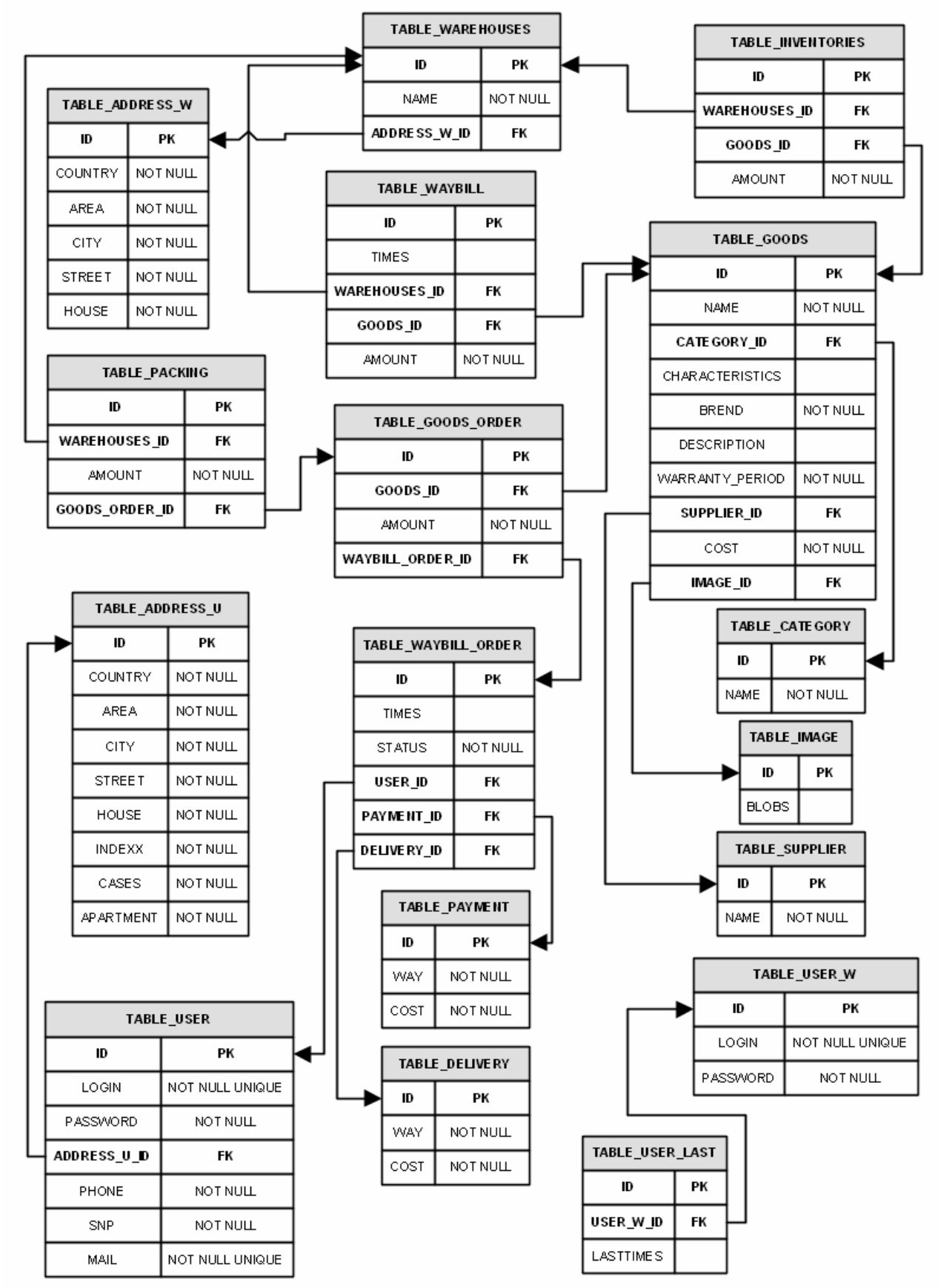
• EndOrder.jsp – страница, на которой клиент может просмотреть статус выполнения заказа (диагностическое сообщение);

• MySection.jsp – персональная страница пользователя;

• GetMoreGoods.jsp – страница, на которой пользователь сможет просмотреть более подробное описание товара.

3.7. Структура БД

Для хранения информации используется база данных под управлением СУБД Oracle 9i. База данных имеет следующую структуру:



В таблице TABLE_USER_W хранятся данные об учетных записях персонала магазина, то есть логин и пароль. Записи в таблице выглядят следующим образом:

ID	LOGIN	PASSWORD
1	USER1	82FD8614A577C46F1A02040E1E45CD64

Таблица 1.

Таблица TABLE_USER_LAST предназначена для хранения времени о последнем входе рабочего. Данная таблица в первую очередь предназначена для того, чтобы администратор мог проверять рабочих магазина, выполняют ли они свои функции в отведенное для них время. Также возможно отследить несанкционированный доступ со стороны посторонних. Если один из рабочих магазина не работал в то время, когда указано в таблице, значит, кто-то вошел под его учетной записью через веб-интерфейс и администратор должен срочно сменить пароль для этого рабочего. Записи в таблице выглядят следующим образом:

ID	USER_W_ID	LASTTIMES
1	1	08.02.2007 12:35

Таблица 2.

Таблица TABLE_ADDRESS_U предназначена для хранения сведений об адресе клиентов для доставки товаров. Записи в таблице выглядят следующим образом:

ID	INDEXX	COUNTRY	AREA	CITY	STREET	HOUSE	CASES	APARTMENT
1	56465	Russia	Samarskaya	Samara	Promishler	190	0	36
2	56465	Russia	Ulyanovckaya	Ulyanovck	Sovetskaya	285	0	88
3	56465	Russia	Tomskaya	Tomsk	Kalugskaya	100	0	96
4	453454	Ро&a	М&#1086	М&#10	М&#	4545	0	18

Таблица 3.

Таблица TABLE_USER предназначена для хранения сведений о личных данных клиента. Записи в таблице выглядят следующим образом:

ID	LOGIN	PASSWORD	MAIL	ADDRESS_U_ID	PHONE	SNP
1	aslan	79FF3A8ADE43678	aslan@google.com	1	31525398	Anrey Genadevich Mohov
2	laden	F8D12C06B5A4BC	laden@google.com	2	51525394	Mihail Genadevich Gigulin
3	liwer	39F89EF83A78C46	liwer@google.com	3	37825394	Sergey Genadevich Gukov
4	RASMUS	D5CCDE20F5EF2F	RASMUS@aslan.com	4	1,231E+09	Гали&

Таблица 4.

Таблица TABLE_WAYBILL_ORDER предназначена для хранения сведений о заказах клиентов. Записи в таблице выглядят следующим образом:

ID	TIMES	STATUS	USER_ID	PAYMENT_ID	DELIVERY_ID
1	08.02.2007 13:09	0	1	1	1

Таблица 5.

Таблица TABLE_PAYMENT предназначена для хранения сведений о способах оплаты заказов. Записи в таблице выглядят следующим образом:

ID	WAY	COST
1	Cash-to the courier	150,78
2	Cash on delivery	50,17
3	The advance payment	0
4	Credit card	0
5	Electronic money	0

Таблица 6.

Таблица TABLE_DELIVERY предназначена для хранения сведений о способах доставки заказов. Записи в таблице выглядят следующим образом:

ID	WAY	COST
1	The courier	78,5
2	Packet by rail in territory of Russia	59,22
3	Packet by air mail in territory of Russia	65
4	The custom-made parcel post by air mail in territory of other states	89,5

Таблица 7.

Таблица TABLE_GOODS_ORDER предназначена для хранения сведений о том, сколько каждого товара в заказе. Записи в таблице выглядят следующим образом:

ID	GOODS_ID	AMOUNT	WAYBILL_ORDER_ID
1	1	30	1
2	2	15	1

Таблица 8.

Таблица TABLE_PACKING предназначена для хранения сведений о том, сколько каждого товара с какого склада необходимо взять, чтобы выполнить заказ. Записи в таблице выглядят следующим образом:

ID	WAREHOUSES_ID	AMOUNT	GOODS_ORDER_ID
1	4	0	1
2	3	7	1
3	1	22	1
4	2	1	1
5	1	7	2
6	2	3	2
7	4	0	2
8	3	5	2

Таблица 9.

Таблица TABLE_INVENTORIES предназначена для хранения сведений о том, сколько каждого товара находится на каждом складе в текущий момент. Записи в таблице выглядят следующим образом:

ID	WAREHOUSES_ID	GOODS_ID	AMOUNT
1	3	1	0
2	1	1	0
3	2	1	8
4	3	2	40
5	1	2	0
6	2	2	0
7	4	1	0
8	4	2	0
9	3	3	0
10	4	3	0
11	1	3	0
12	2	3	0
13	3	4	0
14	4	4	0
15	1	4	0
16	2	4	0
17	3	5	0
18	4	5	0
19	1	5	0
20	2	5	0

Таблица 10.

Таблица TABLE_WAYBILL предназначена для хранения сведений о накладных прибытия товаров. Записи в таблице выглядят следующим образом:

ID	TIMES	WAREHOUSES_ID	GOODS_ID	AMOUNT
1	08.02.2007 12:04	1	1	5
2	08.02.2007 12:04	1	2	7
3	08.02.2007 12:04	2	1	9
4	08.02.2007 12:04	2	2	3
5	08.02.2007 12:04	3	1	7
6	08.02.2007 12:04	1	1	17
7	08.02.2007 12:04	3	2	45

Таблица 11.

Таблица TABLE_WAREHOUSES предназначена для хранения сведений о накладных прибытия товаров. Записи в таблице выглядят следующим образом:

ID	NAME	ADDRESS_W_ID
1	Warehouse #1	1
2	Warehouse #2	2
3	Warehouse #3	3
4	Warehouse #4	4

Таблица 12.

Таблица TABLE_ADDRESS_W предназначена для хранения сведений об адресах складов предприятия. Записи в таблице выглядят следующим образом:

ID	COUNTRY	AREA	CITY	STREET	HOUSE
1	Russia	The Samara area	Samara	Kievskaya	1001
2	Russia	The Samara area	Samara	Kievskaya	1002
3	Russia	The Samara area	Samara	Kievskaya	1003
4	Russia	The Samara area	Samara	Kievskaya	1004

Таблица 13.

Таблица TABLE_GOODS предназначена для хранения сведений о товарах. Записи в таблице выглядят следующим образом:

ID	NAME	CATEGORY_ID	CHARACTERISTICS	BREND	DESCRIPTION	WARRANTY_PERIOD	SUPPLIER_ID	COST	IMAGE_ID
1	Digital IXUS 700	1	<CLOB>	Canon	<CLOB>	12	1	12446,64	1
2	EXILIM EX-Z750	1	<CLOB>	Casio	<CLOB>	12	2	13633,29	2
3	FinePix F10	1	<CLOB>	FUJIFILM	<CLOB>	12	3	10257,93	3
4	Pentium 4 550	6	<CLOB>	Intel	<CLOB>	12	2	7647,3	4
5	Athlon64 3000+ S939	6	<CLOB>	AMD	<CLOB>	12	4	3559,95	5

Таблица 14.

Таблица TABLE_CATEGORY предназначена для хранения сведений о категориях товаров. Записи в таблице выглядят следующим образом:

ID	NAME
1	Digital chambers
2	Pocket personal computers
3	Cellular telephones
4	Flash-memory
5	Videocards
6	Processors
7	Winchesters
8	DVD-drives

Таблица 15.

Таблица TABLE_SUPPLIER предназначена для хранения сведений о поставщиках товаров. Записи в таблице выглядят следующим образом:

ID	NAME
1	Eqaulins
2	Minisota
3	Drivston
4	Jindison
5	Aston

Таблица 16.

Таблица TABLE_IMAGE предназначена для хранения изображений товаров.

Отметим несколько важных моментов при проектирование таблиц.

Изображения будут храниться в поле, тип которого – blob (большой бинарный объект).

Для хранения чисел и вещественных чисел необходимо учитывать ряд факторов. Oracle поддерживает тип данных Number(A,B). То есть A – количество цифр, B – количество цифр из A, приходящихся на дробную часть. Именно этот тип мы и будем использовать как для хранения целого, так и дробного.

В языке Java максимальное число бит, хранящихся в переменной типа Long равно 63. В десятичной форме этот размер равен L=263-1=9223372036854775807 (19 цифр). Если задать размер поля равным 19,

то не избежать ошибок, связанных с преобразованием строки в целое при выполнении программы. Поэтому придется задать максимальный размер поля длиной в 18 цифр, что составит максимум 999999999999999999. Этого число хватит как для хранения количества товара на складах, так и для номеров телефонов. Итак, поле этого типа должно быть объявлено так: Number(18).

Для хранения же переменных типа Integer размер поля должен быть равен 9 цифрам, т.к. I=231-1=2147483647 вмещает 10 цифр, но число большее этого всего лишь на единицу вызовет ошибку переполнения. Максимум число типа Integer будет 999999999. Итак, поле этого типа должно быть объявлено так: Number(9).

Дробно число типа Float имеет максимальный размер 3.4028235E38, поэтому при выборе размера поля можно исходить из условий поставленной задачи. Для хранения денежных платежей должно хватить размера (20,2). Итак, поле этого типа должно быть объявлено так: Number(20,2).

Для хранения строк шифрованного пароля мы должны использовать поле с типом Raw. Размер его зависит от длины пароля (в нашем случае это 55 байт или 55 ASCII-символов). Один элемент поля типа Raw имеет размер 4 бита. Соответственно, два элемента имеют размер 1 байт. Шифрованный пароль длиной в максимум 55 символов занимает 56 байтов. После несложных расчетов, находим, что размер поля Raw должен быть равен 56*2=112.

Хранение русских букв в таблицах вызывает следующие трудности – 1 кириллический символ в ISO-кодировке занимает 7 байт, то есть размер поля для хранения строк должен быть увеличен в 7 раз. Например, если строка из 200 латинских букв занимает 200 байт, то таких же 200 кириллических букв будет занимать уже 7*200=1400 байт.

3.8. Реализация многопользовательского доступа

В СУБД Oracle существуют, по крайней мере, два способа обеспечить многопользовательский доступ.

Первый способ основан на том, что перед редактированием какого-либо ресурса базы данных, пользователь должен спросить у системы разрешение на редактирование. Система может ответить ему либо да, либо нет. Если ресурс не занят, то система блокирует его до вызова COMMIT в текущей сессии данного пользователя. Реализуется это таким образом:

```
COMMIT;
```

```
SELECT * FROM TEST FOR UPDATE NOWAIT;
UPDATE TEST SET NAME='NAME' WHERE ID=1;
COMMIT;
```

Если до завершения второго COMMIT происходит такой же запрос в другой сессии другим пользователем, то последнему выводится сообщение:

ORA-00054: указан занятый ресурс и его получение с опцией NOWAIT

На java это реализуется так:

```
try
{
  …
  //JDBC код запроса данных, используя FOR UPDATE NOWAIT
  …
}
catch (SQLException e)
{
   if (e.getErrorCode()==00054)
         System.out.println("Данные редактируются!");
}
```

Если нам необходимо отловить это исключение в PLSQL коде, то необходимо сделать так:

```
e EXCEPTION;
PRAGMA EXCEPTION_INIT(e, -00054);
BEGIN
… -- Запрос данных с использованием FOR UPDATE NOWAIT
EXCEPTION
when e then  --Ресурс занят - отправляем сообщение пользователю
…;
END;
```

Но Oracle рекомендует полностью изолировать транзакцию с помощью установки атрибутов текущей транзакции. Если не вдаваться в подробности, то это на PLSQL выглядит так:

```
COMMIT;
```

```
SET TRANSACTION ISOLATION LEVEL
                                    SERIALIZABLE NAME 'TEST';
…
UPDATE TEST SET NAME='JAVA' WHERE ID=1;
…
COMMIT;
```

В другом сеансе пользователь, использующий любые другие атрибуты транзакций при попытке выполнить обновление строки с идентификатором равным единице, уйдет в бесконечное ожидание до тех пор, пока в главной транзакции не будем выполнен COMMIT или ROLLBACK. Второй пользователь получит сообщение об ошибке:

```
ORA-08177: не могу преобразовать в последовательный доступ для этой транзакции
```

Отлавливать исключения стоит также как и при первом варианте.

На языке Java использовать данную возможность можно с помощью конструкции:

```
conn.setTransactionIsolation(conn.TRANSACTION_SERIALIZABLE);
```

где conn – переменная типа Connection для соединения с БД.

3.9. Реализация модулей PLSQL

Для создания логики на уровне базы данных придется писать модули на встроенном процедурном языке под названием PLSQL.

Для функционирования системы были написаны две процедуры и пятнадцать функций. Рассмотрим каждую поподробнее.

Процедура GET_INFO_CLIENT_GOODS

Модуль создан специально для показа пользователю статуса магазина. Если клиентов будет зарегистрировано много, то приоритет будет высоким и клиент обязательно совершит покупки. Количество товаров так же будет внушать доверие.

Данный модуль возвращает количество наименований товаров и количество зарегистрированных клиентов.

Процедура UPDATE_USER_LAST

Модуль обновляет дату и время последнего входа рабочего для совершения тех или иных операций с базой данных.

Функция REGISTRATION

С помощью данного модуля можно зарегистрировать клиента в магазине. Если операция происходит успешно, то возвращается идентификатор зарегистрированного пользователя, иначе – ноль.

Функция AUTH_USER

С помощью данного модуля клиент авторизуется в магазине. Если авторизация проходит успешно, возвращается идентификатор клиента. В данной функции происходит сравнение зашифрованной строки со строкой, хранящейся в базе данных.

Функция REGISTRATION_NEW_GOODS

С помощью данного модуля можно зарегистрировать новый товар в базе данных. Если регистрация товара проходит успешно, возвращается указатель на поле, хранящее изображение данного товара. Это необходимо для того, чтобы занести файл с изображением в другую таблицу, а в текущей поставить ссылку на данное поле.

Функция FIND_STRING

С помощью данного модуля можно произвести контекстный поиск по товарам. Возвращаются идентификаторы на найденные товары. То есть возвращается строка вида $<ID_1;..;ID_N>$.

Функция GETLASTTIME

С помощью данного модуля можно получить время последнего захода рабочего в магазин.

Функция AUTH

С помощью данного модуля персонал магазина авторизуется в магазине. В данной функции происходит сравнение зашифрованной строки со строкой, хранящейся в базе данных.

Функция ADD_WAREHOUSES

С помощью данного модуля можно зарегистрировать новый склад и все товары, которые есть в магазине автоматически зарегистрировать на

этом складе. Дело в том, что каждый товар должен быть зарегистрирован на всех складах, которые есть у предприятия. Количество каждого товара будет равно 0 на этом складе.

Функция ADD_DELIVERY

С помощью данного модуля можно добавить новый способ доставки заказов.

Функция ADD_PAYMENT

С помощью данного модуля можно добавить новый способ оплаты заказов.

Функция ADD_SUPPLIER

С помощью данного модуля можно добавить нового поставщика товаров.

Функция ADD_CATEGORY

С помощью данного модуля можно добавить новую категорию товаров.

Функция ISSUE_WAYBILL

С помощью данного модуля можно оформить накладную прихода на товар.

Функция CANCEL_REG_PERS_SHOP

С помощью данного модуля администратор может удалить учетную запись персонала магазина.

Функция REGISTRATION_PERSONNEL_SHOP

С помощью данного модуля администратор может регистрировать персонал магазина.

Функция REGISTRATION_ORDER

С помощью данного модуля клиент может оформить заказ. В случае успеха возвращается 0. Если какого-либо товара из заказа не хватает на складе, то возвращается идентификатор этого товара. Данный модуль выполняет довольно большую работу. В его задачи входит изменять

данные о количестве товаров на складах после вычета количества, приходящегося на заказ, внесение характеристик заказа в базу данных. Также этот модуль автоматизирует комплектацию заказов, то есть показывает рабочим, сколько и с какого склада необходимо выбрать товаров, чтобы выполнить заказ.

3.10. Реализация модулей Java

Для решения главной задачи по программированию логики на промежуточном уровне были созданы семнадцать классов, каждый из которых выполнял свою работу. Все модули упакованы в jar-архив Shop.jar.

Модуль ShowImage.java

Приложение представляет собой апплет. Логика его работы заключается в том, чтобы соединиться с веб-сервером, а именно получить от сервлета бинарные данные, представляющие собой изображение товара.

Апплет прорисовывает изображение в определенной области браузера. Ему на вход подаются три параметра: x и y координаты области, в которой будет происходить рисование и идентификатор изображения в БД. Апплет должен находиться в корне веб-директории, т.е. чтобы переменная CodeBase была равна http://myhost/.

Модуль ImageServlet.java

Сервлет для извлечения BLOB-рисунков из БД. Главный метод обрабатывает Get-запросы, поступившие от клиента.

Для соединения использует минимальный набор прав пользователя SHOPMIN.

Модуль Admin.java

Модуль для администратора БД. Основное назначение: регистрировать рабочий персонал для работы в магазине. Дополнительное назначение: вызов методов преобразования файлов *.jsp в ISO-8859-1. Программа предоставляет пять операций:

1 – Добавить пользователя
2 – Удалить пользователя
3 – Создать симметричный секретный ключ
4 – Показать всех пользователей
5 – Преобразование файлов

Предоставляет консольный интерфейс.

Для работы администратор должен ввести логин и пароль, затем произойдет соединение с базой данных.

Модуль CodingISO.java

Модуль кодирования файлов из Win-1251 в ISO-8859-1. Работает как со строками, так и с файлами. В файлах находит буквы кириллицы и преобразует их в ISO кодировку. Один символ кириллицы занимает 7 байт в ISO-кодировке. Например, первая буква "А" имеет формат "А".

Модуль ShopResultSet.java

Модуль предназначен для преобразования набора данных ResultSet в тип List для доступа к строкам и столбцам как к элементам списка. Данные из набора ResultSet представляют собой таблицу M[i,j]. Конечный список будет иметь вид L[1,1]...L[1,i]...L[j,1]...L[j,i]. Таким образом, матрица вытягивается в прямую линию.

Модуль FormHTML.java

Модуль для генерации динамических JSP-страниц для раздела персонала магазина.

Модуль GoodsWarehousesClass.java

Модуль, методы которого позволяют выводить статистику на странице GoodsWarehouses.jsp. Статистические данные уже не раз упоминались в данной курсовой работе.

Модуль ShopCipher.java

Модуль предназначен для шифрации паролей пользователей.

Модуль GoodsEdit.java

Модуль, методы которого предоставляют интерфейс для работы с пунктом с пунктом "Редактирование товаров".

Модуль ShowGoods.java

Модуль, методы которого предоставляют интерфейс для работы с пунктом с пунктом "Просмотр товаров".

Модуль Orders.java

Модуль предназначен для обработки заказов персоналом магазина. Предоставляет функции для работы страниц OrderMain.jsp, OrderOne.jsp и OrderTwo.jsp.

Модуль StorekeeperBean.java

Модуль, представляющий собой главный JavaBean для персонала магазина. Именно его используют все JSP-страницы для выполнения логики.

Работает с правами пользователя SHOP. В тексте этого класса находится пароль в открытом виде. Если хакеру удастся скопировать этот файл, то конфиденциальность данных в базе данных будет обречена на нет.

Модуль HtmlPage.java

Модуль, методы которого генерируют html-код стандартных элементов браузера.

Модуль Basket.java

Модуль, реализующий концепцию виртуальной корзины при посещении магазина. Каждый элемент корзины абстрактно представляет собой запись, состоящую из 4х элементов: идентификатор товара, количество, цена и название товара. Над элементами корзины можно выполнять основные функции: добавлять товар, удалять товар, редактировать количество товара при оформлении заказа. Также возможно просматривать содержимое корзины.

Модуль PLSQLStored.java

Модуль, методы которого представляют собой оболочку для работы с хранимыми объектами БД персонала магазина. Работает через интерфейс JDBC.

Модуль PLSQLStoredUser.java

Модуль, методы которого вызывают хранимые объекты из БД. Процедуры и функции этого класса предназначены для клиентской части.

Модуль ClientBean.java

Модуль предназначен для работы клиента с магазином. Представляет собой "обертку" для других классов, через которые происходит взаимодействие с базой данных. Для отладки выполненных пользователем операций в консоли выводятся сообщения о возможных ошибках. Администратор веб-сервера может просмотреть данные ошибки и своевременно на них отреагировать. Максимальное количество товаров, которое может заказывать пользователь, ограничено ста единицами. Пользователь может заказывать товары только в том случае, если он авторизован.

3.11. Реализация безопасности

Безопасность приложения на первом уровне

Таблицы, хранящиеся в базе данных, должны быть спроектированы таким образом, что доступ к ним был жестко разграничен на уровне самой базы данных.

Поэтому для реализации данной задачи необходимо создать дополнительно трех системных пользователей: SHOP, SHOPUSER, SHOPMIN. Правами первого пользователя будет обладать персонал магазина. Правами второго – клиент. Третий же пользователь нужен для соединения с базой данных для извлечения изображений из базы данных. У него самый минимальный набор прав.

Пароль пользователя SHOP не должны знать посторонние, кроме как персонал магазина. Заметим, что персонал не имеет прав для просмотра даже шифрованных данных о себе, так как таблицы для хранения этих сведений хранятся в базе от имени администратора базы данных SYSTEM. Данный пользователь сможет просматривать шифрованные данные клиентов базы данных.

Права пользователя SHOPUSER очень опасны с точки зрения безопасности. Узнав пароль к этому пользователю, можно соединяться с БД и просматривать сведения обо всех клиентах базы данных.

Напомним, что для работы клиента, так и персонала в системе создается не множество системных пользователей Oracle, а только один – SHOPUSER. Эта мера является вынужденной, так как не имеет смысла заводить учетную запись в базе данных для каждого клиента. То же самое касается и персонала – у них общий пользователь на всех с необходимыми правами в системе.

Отсюда вытекает следствие – пароль к SHOPUSER должен быть закрыт.

Ну а пароль к SHOPMIN может стать достоянием гласности. Так как с его правами можно только соединяться с БД и просматривать изображения товаров.

Зайдя немного вперед можно отметить, что эти три пароля должны храниться в исходном тексте конечного приложения, поэтому со стороны системного администратора необходимо ограничить доступ к файлам с расширением jar и class.

Безопасность приложения на втором уровне

Никто не сомневается, что хранить номера кредитных карт и пароли клиентов в базе данных в открытом виде нельзя. Доступ к БД могут иметь многие пользователи, в частности персонал магазина, работающие на предприятии. Не все так бескорыстны. Поэтому эта мера является вторым шагом по обеспечению приватности данных пользователей в системе.

Отсюда вытекает следствие – программный комплекс должен иметь модель для шифрования данных. Рассмотрим более детально данную проблему.

Интерфейс API для реализации безопасности входит в ядро платформы J2SE. Пакет для работы с данным интерфейсом называется java.security. Классы данного пакета предназначены для того, чтобы позволить разработчикам включать всю функциональность по обеспечению безопасности высокого уровня в свои приложения.

Первый выпуск Security API в JDK 1.1 представлен как «Архитектура криптографии Java» или сокращенно – JCA, структура для получения доступа к криптографической функциональности платформы Java. В JDK 1.1., JCA включал API для работы с цифровыми подписями и цифровыми сообщениями.

В последующих выпусках Java 2 SDK, корпорация Sun расширила свою архитектуру JCA. Произошла модернизация инфраструктуры управления X.509v3 сертификатами. Ознаменовался выпуск новой архитектуры безопасности Java платформы, которая стала более конфигурируема, гибка в реализации проектов и расширения контроля доступа.

Архитектура JCA охватывает часть пакета безопасности Java 2 SDK, связанной с криптографией. Она включает архитектуру «провайдера». Провайдер обеспечивает многократный вызов необходимых методов в одном сеансе работы приложения.

Расширение Java Cryptography (JCE) представляет собой Фреймворк для работы с шифрованием, генерацией ключей и определением подлинности цифровых сообщений. Поддержка симметричного и ассиметрич-

ного шифрования. Поддержка шифрации и дешифрации потоков «на лету» является, несомненно, полезным способом для написания программ, так как для программиста данные операции происходят «прозрачно». Программное обеспечение также поддерживает безопасные потоки и зашифрованные объекты.

JCE был интегрирован в платформу J2SE начиная с релиза 1.4.

В JDK используется провайдер SunJCE, который предоставляет следующие возможности, необходимые в нашем проекте:

- реализация DES (FIPS PUB 46-1), тройного DES (DES-EDE), и Blowfish алгоритмов шифрования;
- ключевые генераторы для создания ключей, подходящих для DES, Triple DES, Blowfish, HMAC-MD5, и HMAC-SHA1 алгоритмов;
- «фабрики» по производству секретных ключей для обеспечения двунаправленного преобразования между непрозрачным представлением ключа DES, DES-EDE и прозрачным представлением его в виде отдельного файла;

JCE входящая в JDK, включает два компонента программного обеспечения, которые мы будем использовать:

- Фреймворк, определяющий и поддерживающий криптографические сервисы. API находится в пакете javax.crypto;
- провайдер по имени «SunJCE».

Выбор способа шифрования

Шифром принято называть обратимый способ преобразования информации с целью защиты ее от просмотра, в котором используется некий секретный элемент. Исходная информация в этом случае будет называться открытым текстом, а результат применения к ней шифра - закрытым текстом.

Шифры бывают блочными и потоковыми. Блочный шифр работает с сообщениями фиксированного размера (например, 64 бита), а поточный – шифрует весь поток данных посимвольно (например, побайтно). Известные блочные шифры – DES, IDEA, Blowfish, потоковые – RC4. Блочность шифра не означает невозможность шифрования им сообщений, превышающих по длине размер блока.

В открытых компьютерных системах большее распространение получили блочные шифры, в то время как классические поточные шифры обычно ориентированы на аппаратную реализацию, являются секретными и используются преимущественно в специализированных системах связи.

По своей природе шифры делятся на симметричные и ассиметричные. Симметричные шифры используют для шифрации и дешифрации один и тот же секретный ключ. В то же время ассиметричные шифры шифруют сообщение по открытому ключу, а расшифровывают по секретному закрытому ключу. Необходимо также отметить, что реализация ассиметричных алгоритмов работает гораздо медленнее, чем симметричных алгоритмов.

Так как нам не придется передавать секретный ключ по сети, то использование ассиметричного ключа для шифрования паролей теряет смысл в нашей задаче.

Остановим свой выбор на блочно-симметричном способе шифрования информации под названием «DES-EDE». Автор алгоритма DES: National Institute of Standards and Technology (NIST). DES – это 64-битный алгоритм с эффективной длиной ключа в 56-bits (хотя часто говорят о 8 байтах, но старший бит в байте не используется). DES-EDE – тройной DES. Алгоритм шифрования состоит в следующем: исходный текст зашифровывается DES с ключом K1, результат расшифровывается DES с ключом K2, а этот результат опять зашифровывается DES с ключом K3. Итого длина эффективная длина ключа получается 168 бит. Действительная длина ключа все равно будет 192 бита или 24 байта.

Использование данного способа шифрования избавляет нас от юридической ответственности, так как данный алгоритм является на данный момент открытым.

Ключи для шифрования

Исходя из соображений безопасности, администратор должен создать два секретных ключа. Первый будет шифровать пароли клиентов, второй будет шифровать пароли персонала магазина. Если Интернет-магазин в дальнейшем будет использовать информацию о кредитных карточках, то необходимо будет создать еще ключ. Задача автора предоставить такую возможность заказчику.

Итак, мы условились о создании двух ключей. Оба ключа должны будут храниться на веб-сервере для проведения задач авторизации, как персонала, так и клиентов магазина. Естественно, эти ключи должны быть скрыты от лишних глаз и очень хорошо защищены. Один из секретных ключей для шифрования паролей персонала, администратор должен иметь при себе, так как он может создавать учетные записи, а это требует, в свою очередь, того чтобы пароль шифровался уже на машине администратора.

Безопасность приложения на третьем уровне

Клиенты, передавая свои данные через Интернет, должны быть уверены в конфиденциальности передаваемых данных. Сейчас стандартом для прозрачного шифрования данных является протокол Ssl. Так как веб-интерфейс является нашей главной задачей, то мы должны предоставить пользователю возможность передавать свои данные через протокол Https (протокол Http, использующий Ssl туннель).

Так как архитектура разрабатываемой системы является трехуровневой, значит, информация от пользователя может пройти как минимум два важных участка на пути к БД.

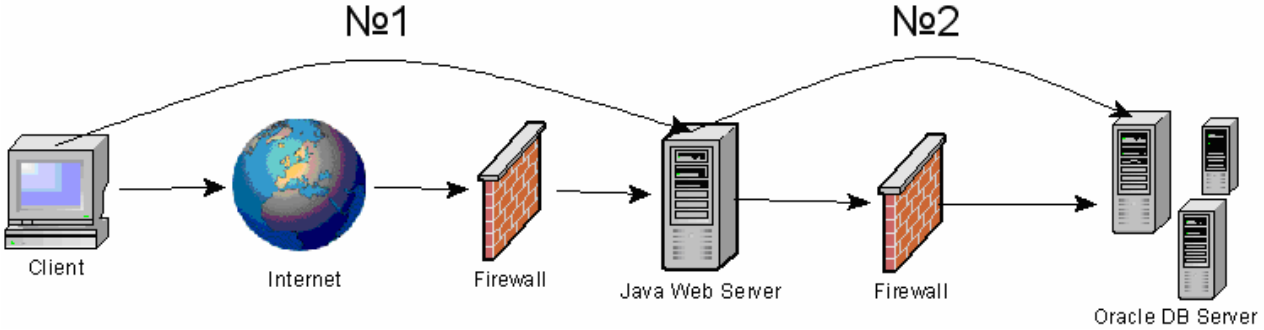


Рис. 7. Получение данных от пользователей

На участке №1 данные шифруются и посылаются на обработку веб-серверу с использованием шифрующего канала Ssl. Участок №2 является внутренней сетью предприятия. На этом участке пароль уже зашифрован с помощью симметричного ключа.

Сертификат для веб-сервера

С первых дней электронной коммерции остро стоял вопрос, как осуществить привязку сервера, сайта, домена к определенному юридическому или физическому лицу? Интернет - настолько свободная информационная среда, что любой может выдать себя за кого угодно.

Для решения этой проблемы были созданы сертификационные центры (Certification Authority). Самые известные из них – Thawte и VeriSign. Эти организации занимаются выдачей электронных сертификатов юридическим и физическим лицам. Сертификат привязан к определенному домену. Использование сертификата на сайте с другим доменом невозможно – клиенту будет выдаваться предупреждение, сообщающее о несоответствии домена, указанного в сертификате реальному. Центры ведут реестр лиц, которым выданы сертификаты, содержащие паспортные данные, ИНН и другие сведения, достаточных, например, для предъявления исков конкретному лицу.

В нашем случае хватит самоподписанного сертификата. Если пользователь не хочет оставлять о себе сведения на сайте, имея перед собой только лишь наш самоподписанный сертификат, то мы можем приобре-

сти сертификат в компании VeriSign. Это фирма является надежным посредником, которому доверяет и руководство магазина и сам клиент. Поэтому сертификат должен быть подписан именно в этой компании.

Для создания и управления сертификатами в пакете JDK предусмотрена специальная программа keytool. Эта программа управляет хранилищами ключей (keystore), базами данных сертификатов и закрытых ключей. Каждый элемент хранилища ключей имеет псевдоним (alias).

Теперь преступим к созданию хранилища ключей (хотя введенные данные будут отображены в сертификате). Необходимо выполнить следующий сценарий:

```
keytool -genkey -keyalg rsa -alias Shopalias -keystore Shopkeystore
```

Утилита попросит ввести пароль для доступа к сертификату, затем данные о хосте, названии организации, компании, городе, области и стране. Затем необходимо подтвердить, набрав yes. После необходимо подтвердить пароль.

Далее в настройках веб-сервера будет показано, как использовать полученный файл.

Если необходимо преобразовать полученный файл к виду, пригодному для обычного просмотра (сертификат, который можно просмотреть в ОС Windows XP), то придется выполнить следующую команду:

```
keytool -export -alias Shopalias -keystore Shopkeystore -file Shopalias.crt
```

Данную команду должен выполнить администратор веб-сервера, так как необходим пароль к хранилищу ключей. Затем сертификат будет иметь расширение crt.

Если в настройках веб-сервера все будет прописано правильно, при заходе клиента на сайт через Https, ему будет выведено окно, в котором он сможет просмотреть данный сертификат и импортировать его в хранилище сертификатов в случае полного доверия сертификату.

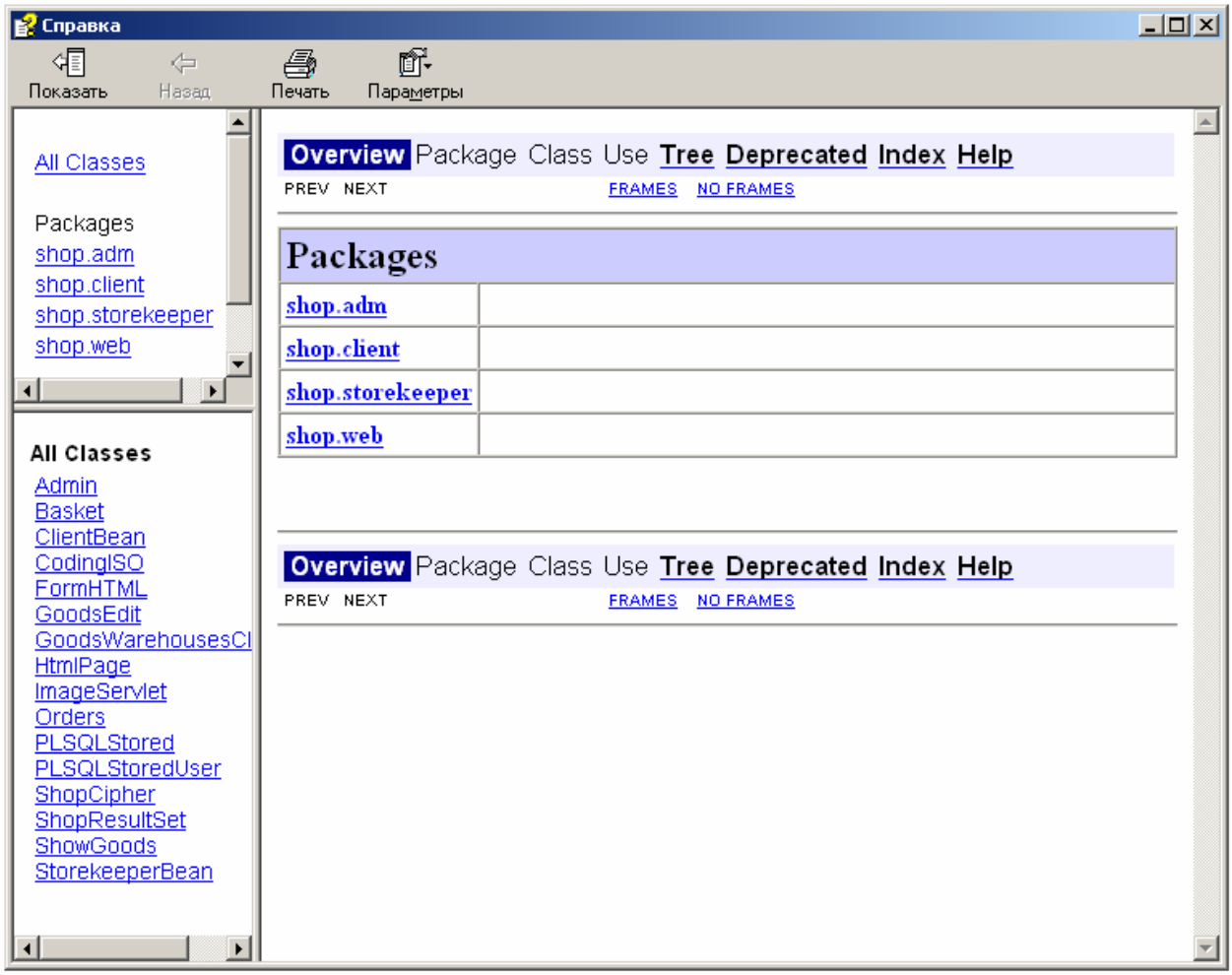
3.12. Дополнительная документация в виде javadoc

Программа javadoc из пакета JDK генерирует документацию API в формате HTML (по умолчанию) для указанных пакетов и классов.

Javadoc использует компилятор javac при работе с указанными исходными файлами и файлами из указанных пакетов. На основе полученной от компилятора информации она создает подробную документацию API. При этом используются все комментарии из исходных файлов. Программу можно применять для создания документации ко всему пакету.

По умолчанию программа javadoc создает документы в формате HTML, но можно определить класс doclet, генерирующий документацию в нужном формате. Вы можете написать собственный класс doclet с помощью doclet API из пакета com.sun.javadoc. Этот пакет описан в стандартной документации к Java 1.2 и последующим версиям.

Итак, мы разработали пакет shop.jar. Полученные классы могут понадобятся в будущем, поэтому в среде NetBeans можно одним кликом мышки запустить javadoc с соответствующими ключами и в папке с проектом появится файлы с public методами всех классов:



ЗАКЛЮЧЕНИЕ

В результате проделанной работы был разработан механизм веб-доступа к базе данных Oracle 9i. Данный механизм позволяет получить доступ к информации Oracle 9i из внешних приложений, например, из скриптовых языков. Это делает возможным интегрировать Oracle 9i с Интернет-технологиями, что в свою очередь позволяет решать широкий круг задач. Примерами таких задач могут являться: создание Интернет-магазина и организация электронной коммерции, веб-доступ к базе данных из удаленных подразделений фирмы и прочее. Следовательно, предприятие, учреждение или организация может существенно расширить сферу своей деятельности, автоматизировать ее, не прибегая к использованию другой учетной системы и работая в рамках одной платформы. Это намного эффективнее и продуктивнее, позволяет автоматизировать деятельность и, следовательно, снизить расходы на ее поддержание.

Работа механизма доступа основана на использовании возможностей технологии JDBC. Очень важно то, что разработка позволяет осуществлять многопоточную работу с данными, хранящимися в базе данных, и механизм транзакций, реализованный в Oracle и JDBC, позволяет это сделать с легкостью. Это необходимо при реализации полноценных веб-приложений, таких как Интернет-магазин.

Автор достиг цели – создать полноценное веб-приложение, реализующее механизм работы Интернет-магазина. На его примере была показана работа механизма доступа к базе данных, а также основные функциональные элементы Интернет-магазина.

Данная разработка – актуальна, так как сервер баз данных Oracle является стандартом для построения больших кластерных и корпоративных систем, реализуем на всех платформах. В то время как все упомянутые технологии Java от Sun – стандарт при построении мощных, распределенных и безопасных Web-приложений.

ИСТОЧНИКИ ИНФОРМАЦИИ

1. Д. Флэнаган, Java. Справочник, 4-е издание – Пер. с англ. – СПб: Символ-Плюс, 2004. – 1040 с., ил.
2. К. Хорстманн, Г. Корнелл, Java 2 Библиотека профессионала, том 2. Тонкости программирования, 7-е изд.: Пер. с англ. – М.: Издательский дом “Вильямс”, 2006. – 1168 с.: ил. – Парал. тит. англ.
3. Н. Моррисо-Леруа, М. Соломон, Д. Басу, ORACLE 8i: Java-компонентное программирование при помощи EJB, CORBA и JSP, Издательство “ЛОРИ”, 2001.
4. С. Урман, Oracle8: Программирование на языке PL/SQL, Издательство “ЛОРИ”, 2001.
5. К. Луни, М. Терьо и эксперты TUSC, Oracle 9i: Настольная книга администратора, Издательство “ЛОРИ”, 2004.
6. В. Будилов, Интернет-программирование на Java. – СПб.: БХВ-Петербург, 2003. – 704 с.: ил.