МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Е.И. Николаев

ТЕХНОЛОГИИ СУПЕРКОМПЬЮТЕРНЫХ КЛАСТЕРОВ И РАСПРЕДЕЛЕННЫХ СИСТЕМ

Методические указания к лабораторным работам

09.04.02 Информационные системы и технологии

Ставрополь, 2026

УДК 004.41 (075.8)
ББК 22.18я73
Н 63

Печатается по решению
учебно-методического совета
Северо-Кавказского федерального
университета

Н 63 **Технологии суперкомпьютерных кластеров и распределенных систем:** учебное пособие (лабораторный практикум) для студентов направления 09.04.02 «Информационные системы и технологии». / Николаев Е.И. – Ставрополь: Изд-во СКФУ, 2026. – 60 с.

Учебное пособие (лабораторный практикум) по дисциплине «Технологии суперкомпьютерных кластеров и распределенных систем» для студентов направления 09.04.02 «Информационные системы и технологии». Пособие охватывает практические аспекты разработки приложений: работа с базами данных, программирование сетевых приложений, использование паттернов проектирования, продвинутые технологии Swing. Предполагается, что студенты, изучающие дисциплину «Технологии суперкомпьютерных кластеров и распределенных систем» обладают базовыми знаниями в области программирования на языке C# или Java, а также основами проектирования и разработки баз данных. Цель второй части учебного пособия: научить студентов решать комплексные задачи в области разработки объектно-ориентированных приложений; обеспечить студентов обширным теоретическим материалом, достаточным для освоения методик разработки приложений; сформировать систему профессиональных компетенций; научить использовать широкий спектр инструментов объектного проектирования и разработки.

УДК 004.41 (075.8)
ББК 22.18я73

Автор:

канд. техн. наук, доцент **Е.И. Николаев**

Рецензенты

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

1. Цели и задачи освоения дисциплины. Учебное пособие (лабораторный практикум) по дисциплине «Технологии суперкомпьютерных кластеров и распределенных систем» нацелено на формирование у студентов практических навыков разработки распределенных корпоративных приложений, которые предполагают совместное использование средств сетевого взаимодействия, технологий баз данных, средств разработки графического интерфейса и теоретических принципов проектирования приложений на основе шаблонов.

Технологии суперкомпьютерных кластеров и распределенных систем, как самостоятельное направление, на текущем этапе развития информационных технологий занимает существенную долю всех доступных средств разработки, анализа, проектирования и моделирования.

Особая роль объектно-ориентированных технологий приводит к необходимости детального изучения принципов построения программных компонент информационных систем на базе объектных технологий. При этом Технологии суперкомпьютерных кластеров и распределенных систем – это только одно из нескольких самостоятельных направлений изучения и использования теории, в основе которой лежат термины «объект» и «класс». При употреблении термина «Технологии суперкомпьютерных кластеров и распределенных систем» подразумевается вся совокупность всех языков программирования, которые по совокупным признакам можно отнести к данной группе. Такие языки относят к объектно-ориентированным языкам программирования (ООЯП). При этом ООП включает также различные технологии программирования, которые используются на практике при разработке приложений на ООЯП.

2. Перечень планируемых результатов обучения по дисциплине (модулю), соотнесённых с планируемыми результатами освоения образовательной программы.

Код, формулировка компетенции	Код, формулировка индикатора	Планируемые результаты обучения по дисциплине (модулю), характеризующие этапы формирования компетенций, индикаторов
УК-1: способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	ИД-1$_{УК-1}$ Определяет полноту информации, степень ее адекватности для решения проблемной ситуации	Применяет принципы определения полноты информации, степени ее адекватности для решения проблемной ситуации
	ИД-2$_{УК-1}$ Анализирует проблемную ситуацию как систему, выявляя ее составляющие и связи между ними	Анализирует проблемную ситуацию как систему, выявляя ее составляющие и связи между ними
	ИД-3$_{УК-1}$ Критически оценивает надежность источников информации; работает с противоречивой информацией из разных источников	Демонстрирует навыки критической оценки надежности источников информации; работает с противоречивой информацией из разных источников
	ИД-4$_{УК-1}$ Вырабатывает стратегию действий для решения проблемной ситуации	Демонстрирует навыки выработки стратегии действий для решения проблемной ситуации
ПК-8: способен управлять проектами и принимать управленческие решения в условиях различных мнений	ИД-1$_{ПК-8}$ Демонстрирует знание принципов управления проектами и принятия управленческих решений в условиях различных мнений	Применяет на практике принципы управления проектами и принятия управленческих решений в условиях различных мнений
	ИД-2$_{ПК-8}$ Управляет проектами и принимает управленческие решения в условиях различных мнений	Управляет проектами и принимает управленческие решения в условиях различных мнений
	ИД-3$_{ПК-8}$ Корректирует информационные системы и программную документацию с учетом различных мнений	Демонстрирует навыки корректировки информационных систем и программную документацию с учетом различных мнений

ЛАБОРАТОРНАЯ РАБОТА 1. УСТАНОВКА JAVA EE SDK

Цели и содержание

Цели:

– закрепить теоретический материал по обзору платформы Java EE;

– научиться устанавливать и администрировать Java EE Server, RDBMS;

– освоить настройку IDE для работы с Java EE SDK.

Методика выполнения

Задание 1.

Установите Java EE SDK & Database Server (MySQL, Oracle SQL or another DBMS), пользуясь указаниями [1].

Выполните связку Java EE Server & Database Server, создав пул соединений через Data Source.

Протестируйте сборку. Ознакомьтесь с документацией, поставляемой с Java EE SDK.

Примечание: отчет должен содержать полный перечень инструкций по установке и запуску сборки.

Контрольные вопросы

1. Характеристики приложений уровня предприятия.
2. Компонентная модель. Базовая компонентная модель.
3. Интерфейсы базовой компонентной модели.
4. Архитектура Java EE приложений.
5. Виды Java EE клиентов.
6. Компоненты Web уровня.
7. Компоненты бизнес-уровня.

8. Контейнеры Web уровня.

9. Контейнеры бизнес-уровня.

10. Аннотации и инжекция зависимости в Java EE приложениях.

Литература

1. Java EE SDK Downloads [электронный документ] :
http://www.oracle.com/technetwork/java/javaee/downloads/index.html

2. Java EE Documentation [электронный документ] :
http://www.oracle.com/technetwork/java/javaee/documentation/index.html

ЛАБОРАТОРНАЯ РАБОТА 2. ПРОСТЕЙШИЕ СЕРВЛЕТЫ

Методика выполнения

Задание 1.

Используя [1,2] напишите два web приложения, каждое состоящее из одного сервлета (отвечающего за начальную страницу приложения). Каждый сервлет просто выводит приветствие. У первого сервлета переопределен метод service(), у второго методы doGet() и doPost().

Указания к выполнению:

1. Имя приложения имеет следующий вид (для работы на терминале)

Lab2_<Номер_задачи><Ник_студента><Код_группы>,

где Номер_задачи – 1 (реализуется service()) или 2 (реализуется doGet/doPost);

Ник_студента – Фамилия студента на английском языке;

Код_группы – код группы {PMI-1201, MO-1201}

Пример:

Lab2_2IvanovPMI-1201

2. Для настройки сервлета используйте IDE и аннотации.

3. Изучите аннотацию @WebServlet

4. В отчете по работе:

а) опишите последовательность создания сервлета

б) приведите примеры кода сервлета

в) приведите описание параметров аннотации @WebServlet

5. Для зачета работы:

а) продемонстрируйте работоспособность приложения;

б) сдайте электронный отчет преподавателю;

в) если работа сдана с нарушением сроков, ответьте на один из вопросов для зачета, выданный преподавателем.

Контрольные вопросы

1. Виды web приложений.
2. Модель запрос-ответ. Механизм отработки запроса в Java EE приложениях.
3. Виды web компонентов.
4. Протокол HTTP: основные понятия и методы.
5. Состав и жизненный цикл web приложения.
6. Технологии Java EE 7 Web Profile.
7. Понятие сервлета. Сравнение технологии сервлетов с CGI.
8. Роль Java Servlet в технологиях web уровня.
9. Состав Java Servlet API.
10. Базовые интерфейсы Java Servlet API.
11. Класс GenericServlet.
12. Классы и интерфейсы для работы с HTTP.

Литература

1. Java EE Tutorial. [Electronic resource] . – Electronic data. [2026]. – Mode of access : https://docs.oracle.com/javaee/7/tutorial/
2. Your First Cup: An Introduction to the Java EE Platform. [Electronic resource] . – Electronic data. [2026]. – Mode of access : https://docs.oracle.com/javaee/7/firstcup/
Java EE

ЛАБОРАТОРНАЯ РАБОТА 3. ДИСПЕТЧЕРИЗАЦИЯ СЕРВЛЕТОВ

Методика выполнения

Задание 1.

Используя [1, 2] и рекомендованную на лекции литературу напишите web приложение, которое запрашивает имя пользователя и выводит приветствие с учетом введенного имени. Первый сервлет, содержит форму с полем, в которое вводится имя пользователя. Форма должна содержать картинку. В первом сервлете происходит диспетчеризация на второй сервлет, который добавляет в ответ приветствие, содержащее имя пользователя.

Задание 2.

Используя [1, 2] и рекомендованную на лекции литературу напишите web приложение, которое реализует конвертор валюты: вводится курс и количество валюты. Web приложение производит перерасчет. Первый сервлет содержит форму с двумя полями, в которые вводится курс и количество. Второй сервлет проводит пересчет. Если количество конвертированной валюты больше 100 единиц, то вывод осуществляется красным цветом полужирного шрифта. В первом сервлете происходит диспетчеризация на второй сервлет.

Указания к выполнению:

1. Имя приложения имеет следующий вид (для работы на терминале)
Lab3_<Номер_задачи><Ник_студента><Код_группы>,
где Номер_задачи – номер задания {1, 2};
Ник_студента – Фамилия студента на английском языке;
Код_группы – код группы
Пример:
Lab3_2IvanovPMI-1201

2. Для зачета работы:

а) продемонстрируйте работоспособность приложения;

б) если работа сдана с нарушением сроков, ответьте на один из вопросов для зачета, выданный преподавателем.

Контрольные вопросы

1. Понятие сервлета. Сравнение технологии сервлетов с CGI.
2. Роль Java Servlet в технологиях web уровня.
3. Состав Java Servlet API.
4. Базовые интерфейсы Java Servlet API.
5. Класс GenericServlet.
6. Классы и интерфейсы для работы с HTTP.
7. Поясните механизм диспетчеризации.
8. Методы RequestDispatcher. Особенности использования методов.

Литература

1. Java EE Tutorial. [Electronic resource] . – Electronic data. [2026]. – Mode of access : https://docs.oracle.com/javaee/7/tutorial/
Java EE
2. Your First Cup: An Introduction to the Java EE Platform. [Electronic resource] . – Electronic data. [2026]. – Mode of access : https://docs.oracle.com/javaee/7/firstcup/

ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 4. ФИЛЬТРЫ И COOKIES

Методика выполнения

Задание 1.

Используя [1,2] напишите web приложение с фильтром, который по умолчанию задает кодировку win-1251. Приложение формируют форму с именем пользователя, при нажатии кнопки подтверждения приложение выводит приветствие пользователю, общее количество посещение сайта и дату-время последнего посещения.

Задание 2.

Используя [1,2] напишите web приложение с фильтром, который в зависимости от времени суток (ночь, день) меняет цветовое оформление страниц сервера и прикрепляя картинку с солнцем или луной. Разработайте минимум две страницы (различные по содержанию), на которых вводится «добавочное» время (которое прибавляется в фильтре к текущему времени на Java EE сервере, для тестирования работы фильтра).

Задание 3.

Используя [1,2] напишите web приложение с фильтром, который в определяет язык web страницы и если язык отличается от русского, то выводится соответствующее сообщение и кнопка «Перевести». Кроме того, если в запросе не указан язык, то по умолчанию в запрос подставляется русский язык и в лог файл пишется соответствующее сообщение.

Приложение должно содержать:

1. Стартовую страницу со ссылками на русскоязычную и, к примеру, англоязычную страницу.
2. Русскоязычная страница
3. Англоязычная страница.

4. Фильтр хранит данные об известных ему кодировках в map.

Указания к выполнению:

1. Используйте методы

public java.lang.String HttpServletRequest.getHeader(java.lang.String name)

public void HttpServletResponse.setHeader(java.lang.String name,

java.lang.String value)

Java EE

Example Code:

…

public void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException

{

…

response.setHeader("Content-Language", "en");

PrintWriter out = response.getWriter();

…

}

2. Для ведение лог файла (который часто используется для отладки и тестирования серверных приложений) используйте следующие технологии и ссылки.

a) Класс java.util.logging.Logger (API

https://docs.oracle.com/javase/8/docs/api/java/util/logging/Logger.ht

ml)

Пример.

package com.och.edu.web;

import java.io.IOException;

import java.util.HashMap;

import java.util.Map;

import java.util.logging.Logger;

import javax.enterprise.context.RequestScoped;

```
import javax.inject.Inject;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
* Servlet implementation class ControllerServlet
*/
@WebServlet("/ControllerServlet")
public class ControllerServlet extends HttpServlet {
private static final long serialVersionUID = 1L;
private Logger logger = Logger.getLogger("com.och.edu.web.Controllerservlet");
@Inject
@RequestScoped
private ErrorMsg errorMsg;
public ControllerServlet() {
super();
}
@Override
protected void service(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
Map<String, String> actions = new HashMap<String, String>();
actions.put("neworder", "/NewOrder.jspx");
actions.put("addorder", "/AddOrderServlet");
Java EE
actions.put("default", "/Start.jsp");
```

```
String action = request.getParameter("action");
logger.info("initial action: " + action);
if((action==null) || (action==""))){action="default";}
String dispatchTo = actions.get(action);
logger.info("dispatchTo: " + dispatchTo);
if (dispatchTo==null){
logger.info("try to access to injected bean");
errorMsg.setMessage("не найден ресурс: " + action);
response.sendError(HttpServletResponse.SC_NOT_FOUND);
}
RequestDispatcher rd =
getServletContext().getRequestDispatcher(dispatchTo);
if (rd == null){
errorMsg.setMessage("ошибка диспетчера, ненайдено : " + dispatchTo);
response.sendError(HttpServletResponse.SC_NOT_FOUND);
} else {
rd.forward(request, response);
}
}
}
```

б) Log4j - система логирования сообщений в Java

http://www.log4j.ru/

http://logging.apache.org/log4j/2.x/

Контрольные вопросы

1. Что такое фильтр?
2. Опишите механизм работы фильтра.
3. Поясните типовые задачи для фильтров.
4. Опишите основные интерфейсы Filter API.

5. Как происходит закрепление фильтров за web компонентами?
6. Что такое сессия? Свойства сессии.
7. Как происходит доступ к сессии и извлечение данных?
8. Как происходит управление сессией?
9. Что такое cookies? Какова структура cookies?
10. Поясните механизм работы с Cookies.

Литература

1. Java EE Tutorial
2. Your First Cup: An Introduction to the Java EE Platform

ЛАБОРАТОРНАЯ РАБОТА 5. JSP

Методика выполнения

Задание 1.

Использование основных конструкций JSP. Используя [1, 2] напишите web приложение на основе JSP:

1. создайте jsp страницу, отвечающую за корень приложения и выводящее единственное слово «Привет» (не забываем про фильтры);

2. доработайте страницу, чтобы она выводила текущие дату и время:

– используйте директиву для импорта java.util.Date;

– используйте скриплет для объявления переменной, хранящей значение даты-времени;

– используйте выражение внутри HTML для вставки текущей даты с предваряющей надписью «Сегодня …»;

Задание 2.

1. Исследование объявлений. Используя [1,2] напишите web приложение на основе JSP:

– объявите две переменных date1 и date2 класса java.util.Date. Переменную date1 объявите через скриплет (<%... %>), а переменную date2 – через объявление (<%!...%>). Инициализируйте при объявлении (Date date1 = new Date();)

– используя jsp синтакс выведите их значения;

– обновите страницу несколько раз. Что происходит и почему?

Задание 3.

Доступ к параметрам в JSP. Используя [1, 2] напишите web приложение на основе JSP:

– приложение строится на основе паттерна MVC2;

– стартовая страница – форма, запрашивающая имя пользователя;

– разработайте jsp страницу, выводящее имя пользователя с использованием javabean;

– обоснуйте выбранный контекст.

Задание 4.

Исследование включений. Используя [1,2] напишите web приложение на основе JSP:

– создайте текстовый файл, содержащий текст «включение в JSP»;

– создайте jsp, включающую выше указанный текст директивой include;

– включите этот же текст, используя действие (action) <jsp:include …>, не забудьте установить атрибут flush=”true”. Убедитесь, что при отображении текст включен дважды;

– измените текст во включенном файле (на сервере). Обновите страницу. Поясните что и почему произошло.

Литература

1. Java EE Tutorial

2. Your First Cup: An Introduction to the Java EE Platform

ЛАБОРАТОРНАЯ РАБОТА 6. EJB

Метдика выполнения

Задание 1.

Разработайте EJB приложение, которое реализует калькулятор на два поля:

– web слой реализуется по паттерну MVC2 (по желанию можно использовать JFS);

– EJB реализует функции калькулятора;

– при реализации использовать инжекцию зависимости.

Задание 2.

Разработайте EJB приложение, которое реализует калькулятор на одно поле (операции происходят с «памятью» калькулятора – как у всех обычных калькуляторов). Калькулятор, помимо сохранения числа в памяти должен еще и помнить Ваше имя:

– web слой реализуется по паттерну MVC2 (по желанию можно использовать JFS);

– EJB реализует функции калькулятора;

– при реализации использовать инжекцию зависимости.

Задание 3.

Разработайте игровое EE приложение. Приложение «загадывает» некоторое число, например, от 1 до 10. Несколько удаленных пользователей пытаются его одновременно отгадать. Выигрывает тот, кто отгадал первым.

Перегрузку приложения можно не предусматривать:

– web слой реализуется по паттерну MVC2 (по желанию можно использовать JFS);

– при реализации использовать инжекцию зависимости.

В электронном отчете по ВСЕМ заданиям:

– дайте обоснование типа EJB;

– при помощи диаграммы UML поясните архитектору приложения и взаимодействие компонентов (использование профиля UML для Java & EJB1 приветствуется).

Контрольные вопросы

1. Дайте определение EJB. Роль EJB в приложениях уровня предприятия
2. Классификация EJB.
3. Методы доступа к EJB (DI &JNDI).
4. Business Interface & No-interface views.
5. Типы клиентов EJB: local, remote, web service.
6. Программирование local доступа.
7. Программирование remote доступа.
8. Жизненный цикл Stateless bean.
9. Жизненный цикл Stateful bean.
10. Жизненный цикл Singleton bean.
11. Тестирование EJB.

Литература

1. Java EE Tutorial. https://docs.oracle.com/javaee/7/tutorial/doc/home.htm
2. Your First Cup: An Introduction to the Java EE Platform. https://docs.oracle.com/javaee/7/firstcup/doc/home.htm
3. Java EE Documentation. https://docs.oracle.com/javaee/
4. Real World Java EE Patterns: Rethinking Best Practices / Adam Bien. press.adam-bien.com
5. EJB 3.1 Cookbook. / Richard M. Reese. Packt Publishing. 2013.
6. EJB 3 in Action: Second Edition // D. Panda, R. Rahman, R. Cuprak,

M. Remijan, 2014.

7. Professional Java EE Design Patterns // V.Yener, A. Theedom. John Wiley & Sons, 2015.

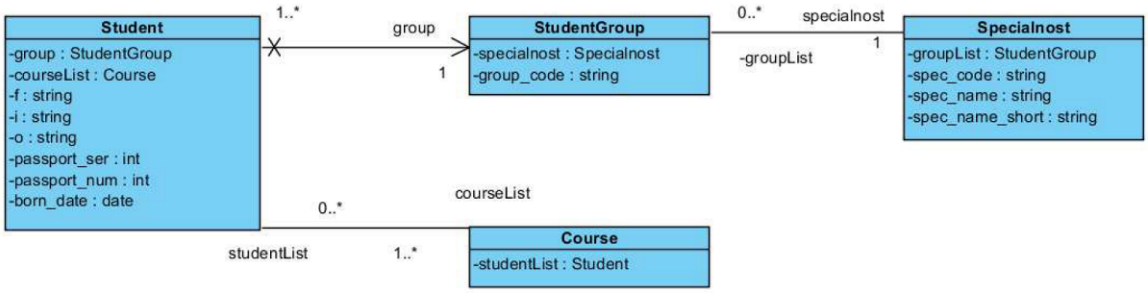
8. Beginning EJB 3. Java EE 7 Edition // J. Wetherbee, C/ Rathod, R. Kodali, P. Zadrozny. Apress.

ЛАБОРАТОРНАЯ РАБОТА 7. JPA

Методика выполнения

Задание 1.

Разработайте Java EE приложение «Деканат» с модулем JPA, реализующим следующую схему сущностей:



Содержание этапов разработки:

а) определите первичные и альтернативные ключи сущностей, доработайте перечень методов сущностей;

б) разработайте веб страницу для отображения перечня специальностей;

в) разработайте веб страницу для отображения перечня групп по выбранной специальности;

г) разработайте веб страницу для отображения перечня студентов по группе (реализовать ленивую загрузку);

д) разработайте веб страницу для отображения перечня курсов по выбранному студенту.

Контрольные вопросы

1. Понятие и виды персистентности данных.
2. Проблема потери соответствия. Системы ORM.
3. Основные понятия JPA. Взаимодействие компонентов в JPA.
4. JPA сущности: понятие, объявление, аннотации.

5. Вложенные сущности.

6. Первичные ключи JPA сущностей.

7. Использование коллекций и валидации в JPA сущностях.

8. Связи между JPA сущностями.

9. Наследование в JPA. Отображение наследования в БД.

10. Управление сущностями через менеджеры сущностей.

11. Язык JPQL.

12. Criteria API.

13. Уровни блокировок при использовании JPA.

Литература

1. Java EE Tutorial. https://docs.oracle.com/javaee/7/tutorial/doc/home.htm

2. Your First Cup: An Introduction to the Java EE Platform. https://docs.oracle.com/javaee/7/firstcup/doc/home.htm

3. Java EE Documentation. https://docs.oracle.com/javaee/

4. Real World Java EE Patterns: Rethinking Best Practices / Adam Bien. press.adam-bien.com

5. EJB 3.1 Cookbook. / Richard M. Reese. Packt Publishing. 2013.

6. EJB 3 in Action: Second Edition // D. Panda, R. Rahman, R. Cuprak, M. Remijan, 2014.

7. Professional Java EE Design Patterns // V.Yener, A. Theedom. John Wiley & Sons, 2015.

Java EE

8. Beginning EJB 3. Java EE 7 Edition // J. Wetherbee, C/ Rathod, R. Kodali, P. Zadrozny. Apress.

ЛАБОРАТОРНАЯ РАБОТА 8. АРХИТЕКТУРА JAVA EE

Теоретическое обоснование

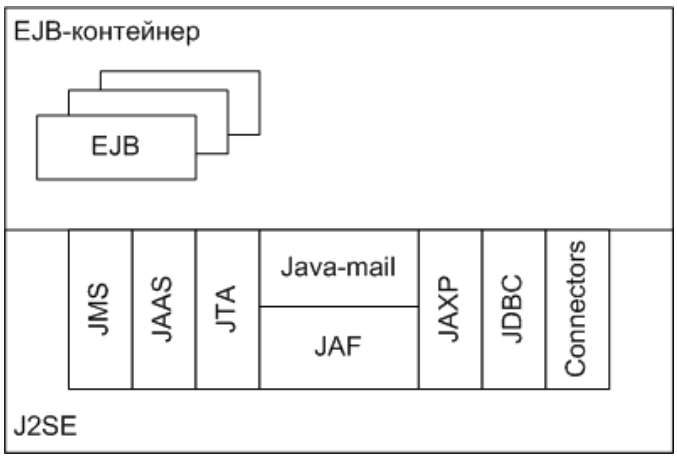
Поддерживаются разные типы клиентов: HTML – браузеры, апплеты, автономные java-приложения.

Уровень представления – часто реализуется в виде веб-уровня.

Уровень бизнес-логики – в виде уровня EJB (Enterprise Java Beans).

Уровень интеграции – уровень сервера БД – EIS (Enterprise Information Server). Это адаптеры ресурсов J2EE.

Сервер приложений – содержит контейнеры компонентов EJB.



Особенности:

- Доступ к инфраструктуре J2EE.
- Управление жизненным циклом компонентов EJB.
- Доступ к БД с использованием JDBC.
- Контейнер изолирует компонент от клиента. Все запросы перехватываются контейнером.
- У каждого компонента есть объект EJBContext, который является ссылкой на контейнер.
- Контейнер автоматически создает набор соединений с БД.
- Контейнер позволяет объединять несколько компонент внутри одной транзакции.

Аббревиатуры:

JMS – Java Messaging Service

JSP – Java Server Page

JTA – Java Transaction API

JAF – Java Beans Activation Framework

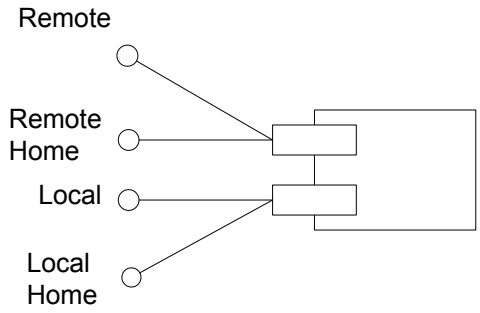
JAXP – Java API for XML Parser

JAAS – Java Authentication and Authorization Service

EJB – Enterprise Java Beans

EJB – серверная java технология, основанная на транзакциях. Позволяет быстро и относительно просто разрабатывать распределенные, транзакционные, безопасные и портируемые Java приложения.

Компонент EJB представляет собой:

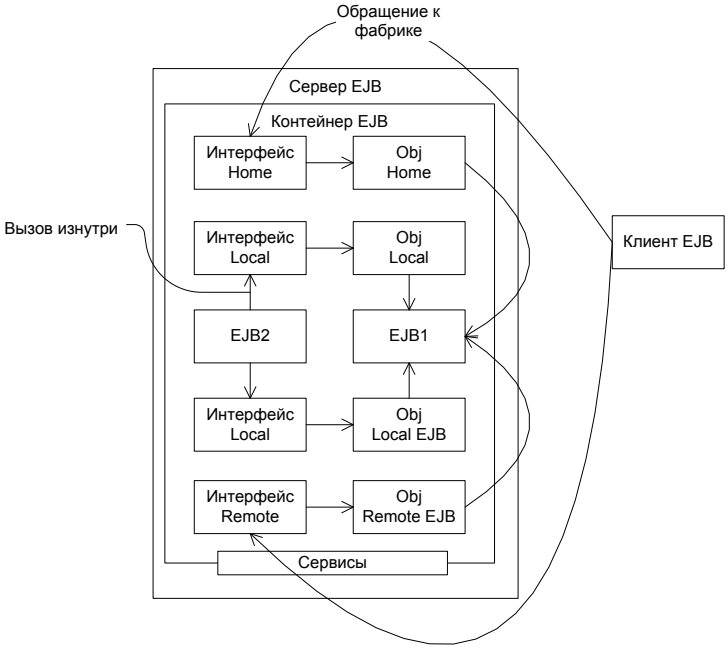


Remote – Расширенный интерфейс. Определяет методы компонента.

Remote Home – определяет методы жизненного цикла для создания, удаления, поиска компонент(интерфейс фабрики классов)

Local – этот интерфейс используется другими компонентами находящимися в этом же контейнере.

Вызов происходит следующим образом



Модули EJB – объединенные в группу компоненты EJB, которые могут взаимодействовать.

Типы компонентов EJB:

Session – связаны с бизнес процессами приложения; имеют доступ к бд, но не предоставляют доступа к ней; жизненный цикл – до перезагрузки сервера. (вызов сессионных компонентов: сервлетты, страницы JSP, java приложения). Разделяется на 2 типа:

Stateless – не сохраняет информации о своем состоянии

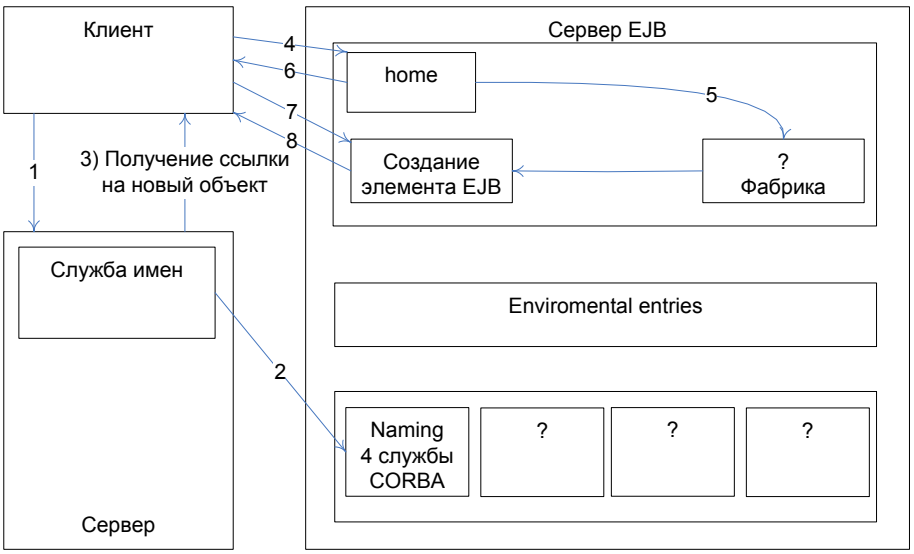
Statefull – могут сохранять инф о своем состоянии

(У них сильно различаются жизненные циклы.)

Entity – моделируют бизнесс данные приложения; предоставляют доступ к БД; часто 1 обращается к 2; t жизни = t жизни бд(при перезагр сервера автоматически восстанавливаются); вызов из 1 и компонентов WEB;

MessageDriven – прдставляют действия. Их можно вызвать только послав сообщение этому компоненту; С помощью 3 организуют доступ к 1. t жизни как у 1

Так цепочку обращений в J2EE можно представить следующим образом:



Java Beans

JB это не EJB, EJB более обширное понятие.

JB – для создания пользовательского интерфейса, для взаимодействия между страницами.

EJB – для создания серв приложений, только не визуальные компоненты.

Методика выполнения

Разработать корпоративное приложение на основе технологии Java EE в соответствии с вариантом. Данные и логика приложения должны находиться на стороне сервера. На стороне клиента может быть как только браузеная страница (pure HTML), так и толстый клиент.

Вариант	Задание

1	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

2	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.

3	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.

4	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.

5	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.

6	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.

7	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.

8	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.

9	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.

10	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.

11	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

12	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.

13	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.

14	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.

15	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.

16	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.

17	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».

18	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».

19	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк): 1 Молоко \ Продукты \ 40.00 \ 1 2 Тряпка \ Хозтовары \ 10.00 \ 5 3 Порошок \ Хозтовары \ 750.00 \ 10 4 Мясо \ Продукты \ 350 \ 5 5 Швабра \ Хозтовары \ 900.00 \ 5 Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.
20	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк): 1 Рис 1 кг. / 120.11 / 54 / Магнит 2 Ручка шариковая / 23.00 / 15 / ТЦ Карандаш 3 Тетрадь 12 л. / 15.00 / 200 / ГеоКнига 4 Тетрадь 60 л. / 90.70 / 300 / ГеоКнига 5 Торт 1 кг. / 450 / 4 / Магнит Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

Контрольные вопросы

1. Понятие и виды персистентности данных.
2. Проблема потери соответствия. Системы ORM.
3. Основные понятия JPA. Взаимодействие компонентов в JPA.
4. JPA сущности: понятие, объявление, аннотации.
5. Вложенные сущности.
6. Первичные ключи JPA сущностей.
7. Использование коллекций и валидации в JPA сущностях.
8. Связи между JPA сущностями.
9. Наследование в JPA. Отображение наследования в БД.
10. Управление сущностями через менеджеры сущностей.
11. Язык JPQL.
12. Criteria API.
13. Уровни блокировок при использовании JPA.

Литература

1. Java EE Tutorial. https://docs.oracle.com/javaee/7/tutorial/doc/home.htm
2. Your First Cup: An Introduction to the Java EE Platform. https://docs.oracle.com/javaee/7/firstcup/doc/home.htm
3. Java EE Documentation. https://docs.oracle.com/javaee/
4. Real World Java EE Patterns: Rethinking Best Practices / Adam Bien. press.adam-bien.com
5. EJB 3.1 Cookbook. / Richard M. Reese. Packt Publishing. 2013.
6. EJB 3 in Action: Second Edition // D. Panda, R. Rahman, R. Cuprak, M. Remijan, 2014.
7. Professional Java EE Design Patterns // V.Yener, A. Theedom. John Wiley & Sons, 2015.
Java EE
8. Beginning EJB 3. Java EE 7 Edition // J. Wetherbee, C/ Rathod, R. Kodali, P. Zadrozny. Apress.

ЛАБОРАТОРНАЯ РАБОТА 9. НЕПОСРЕДСТВЕННОЕ СОЕДИНЕНИЕ С БАЗОЙ ДАННЫХ

Теоретическое обоснование

Для работы с базами данных (БД) в данной лабораторной работе применяется самая простая технология на основе баз данных Apache Derby. Данный инструментарий устанавливается вместе с JDK, но при его отсутствии его можно установить и настроить самостоятельно с использованием ресурса http://db.apache.org/derby.

Перед программированием приложений с использованием баз данных необходимо освоить теоретические основы использования БД в программах Java. Предполагается, что изучающие данный курс владеют основами проектирования и разработки баз данных.

Создатели Java с самого начала осознавали потенциальные преимущества данного языка для работы с базами данных. С 1995 года они начали работать над расширением стандартной библиотеки Java для организации доступа к базам данных средствами SQL. Сначала они попробовали создать такие расширения Java, которые позволили бы осуществлять доступ к произвольной базе данных только средствами Java. Но очень скоро они убедились в бесперспективности такого подхода, поскольку для доступа к базам данных применялись самые разные протоколы. Кроме того, поставщики программного обеспечения баз данных были весьма заинтересованы в разработке на Java стандартного сетевого протокола для доступа к базам данных, но при условии, что за основу будет принят их собственный сетевой протокол.

В конечном итоге поставщики баз данных и инструментальных средств для доступа к ним сошлись на том, что лучше предоставить прикладной интерфейс API только на Java для доступа к базам данных средствами SQL, а также диспетчер драйверов, который позволил бы подключать к базам драйверы независимых производителей. Такой подход позволял поставщикам

баз данных создавать собственные драйверы, которые подключались бы с помощью данного диспетчера. Предполагалось, что это будет простой механизм регистрации драйверов.

Похожая ситуация складывалась при развитии технологии ADO.NET в рамках .NET Framework. В технологии от Microsoft существуют понятия «поставщик данных», «адаптер».

Организация прикладного интерфейса JDBC основана на успешной модели интерфейса ODBC, разработанного в корпорации Microsoft. В основу JDBC и ODBC положен общий принцип: программы, написанные в соответствии с требованиями прикладного интерфейса API, способны взаимодействовать с диспетчером драйверов JDBC, который, в свою очередь, использует подключаемые драйверы для обращения к базе данных. Это означает, что для работы с базами данных в прикладных программах достаточно пользоваться средствами JDBC API.

Каждый драйвер JDBC принадлежит к одному из перечисленных типов:

1. Драйвер типа 1. Преобразует JDBC в ODBC и для взаимодействия с базой данных использует драйвер ODBC. Один такой драйвер был включен в младшие версии Java под названием мост JDBC/ODBC. Но для его применения требуется установить и настроить соответствующим образом драйвер ODBC. В первом выпуске JDBC этот мост предполагалось использовать только для тестирования, а не для применения в рабочих программах. В настоящее время уже имеется достаточное количество более удачных драйверов, поэтому пользоваться мостом JDBC/ODBC не рекомендуется.

2. Драйвер типа 2. Разрабатывается преимущественно на Java и частично на собственном языке программирования, который используется для взаимодействия с клиентским прикладным интерфейсом API базы данных. Для применения такого драйвера, помимо библиотеки Java, на стороне клиента нужно установить специфический для данной платформы код.

3. Драйвер типа 3. Разрабатывается только на основе клиентской библиотеки Java, в которой используется независимый от базы данных

протокол передачи запросов базы данных на сервер. Этот протокол приводит запросы базы данных в соответствие с характерным для нее протоколом. Развертывание прикладных программ значительно упрощается благодаря тому, что код, зависящий от конкретной платформы, находится только на сервере.

4. Драйвер типа 4. Представляет собой библиотеку, написанную только на Java, для приведения запросов JDBC в соответствие с протоколом конкретной базы данных.

Большинство поставщиков баз данных предоставляют драйверы типа 3 или 4. Кроме того, целый ряд сторонних производителей специализируется на создании драйверов, которые позволяют добиться более полного соответствия принятым стандартам, поддерживают большее количество платформ, обладают более высокой производительностью или надежностью, чем драйверы, предлагаемые поставщиками баз данных.

Основные цели прикладного интерфейса JDBC можно сформулировать следующим образом:

1. Разработчики пишут программы на Java, пользуясь для доступа к базам данных стандартными средствами языка SQL (или его специализированными расширениями), но следуя только принятым в Java соглашениям.

2. Поставщики баз данных и инструментальных средств к ним предоставляют драйверы только низкого уровня. Это дает им возможность оптимизировать драйверы под свою конкретную продукцию.

Согласно традиционной модели «клиент-сервер» графический пользовательский интерфейс реализуется на стороне клиента, а база данных располагается на стороне сервера (рис. 9.1). В этом случае драйвер JDBC развертывается на стороне клиента.

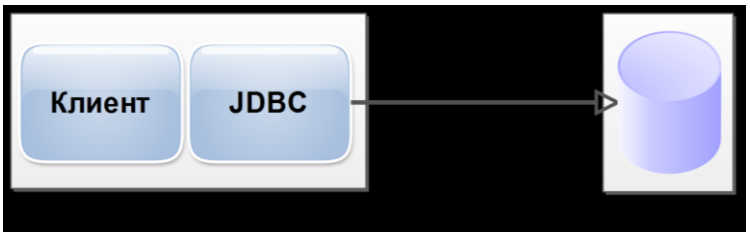


Рисунок 9.1 – Схема приложения типа «клиент-сервер».

В настоящее время существует явная тенденция к переходу от архитектуры «клиент-сервер» к трехуровневой модели или даже более совершенной n-уровневой модели (multi tiers application java). В трехуровневой модели клиент не формирует обращения к базе данных. Вместо этого он обращается к средствам промежуточного уровня на сервере, который, в свою очередь, выполняет запросы к базе данных. Трехуровневая модель обладает двумя преимуществами: отделяет визуальное представление (на клиентском компьютере) от бизнес-логики (на промежуточном уровне) и исходных данных (хранящихся в базе данных). Таким образом, становится возможным доступ к тем же самым данным по таким же бизнес-правилам со стороны разнотипных клиентов, в том числе прикладных программ на Java, аплетов или веб-форм.

Прикладной интерфейс JDBC позволяет взаимодействовать с базами данных с помощью языка SQL, который, в свою очередь, образует интерфейс для большинства современных реляционных баз данных. Настольные базы данных предоставляют графический интерфейс, который дает пользователям возможность непосредственно манипулировать данными, но доступ к серверным базам данных возможен только с помощью SQL.

Пакет JDBC можно рассматривать лишь как прикладной интерфейс API для взаимодействия с командами и Операторами языка SQL для доступа к базам данных.

Для установления соединения с базой данных необходимо указать ряд параметров, имеющих отношение к базам данных. К их числу могут относиться имена хостов, номера портов, а также имена баз данных. В JDBC используется синтаксис описания источника данных, подобный обычным URL. Ниже приведены примеры такого описания.

jdbc:derby://localhost:1527/COREJAVA;create=true

или

jdbc:derby:C:\\Users\\NotDeveloper\\MyDB;create=true

Вам нужно будет также получить JAR-файл, в котором находится драйвер для вашей базы данных. Так, если вы пользуетесь базой данных Derby, вам понадобится файл derbyclient.jar. Если же это другая база данных, вам придется найти для нее подходящий драйвер.

При запуске программы, обращающейся к базе данных, в командной строке нужно указать JAR-файл драйвера после параметра -classpath. Для компиляции самой программы JAR-файл драйвера не нужен.

В среде Eclipse возможно сконфигурировать базу данных, создать подключение, создать таблицы и поля таблиц в визуальном режиме. На рис. 9.2 представлен внешний вид окна «Data Source Explorer», в котором подключена новая БД «MyDB».

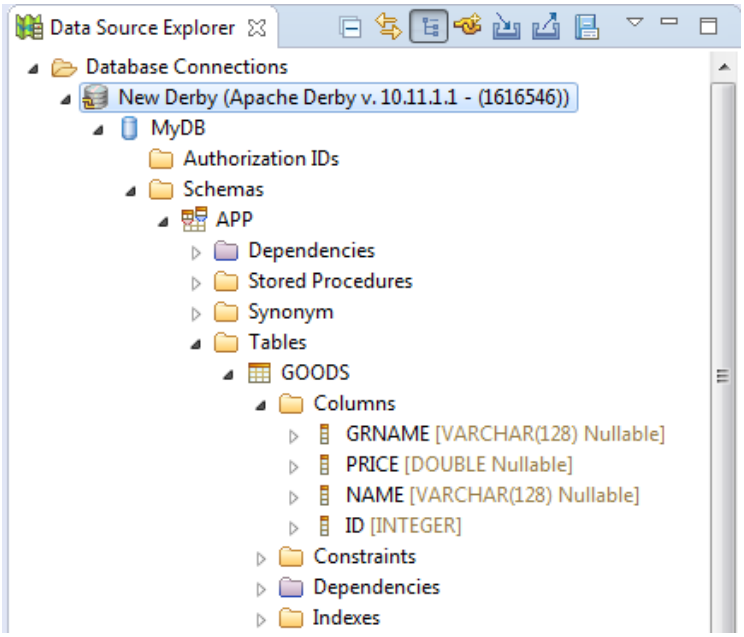


Рисунок 9.2 – Создание источника данных с использованием инструмента «Data Source Explorer».

На рисунке показано, что создана пользовательская таблица, а также поля таблицы (отображаются типы полей).

На рис. 9.3 показан вид окна «Свойства», которое открывается соответствующей командой контекстного меню для соединения «New Derby».

После того как спроектирована база данных и настроено соединение, можно начинать проектировать приложение для работы с базой данных.

В технологии Java перед подключением к БД необходимо зарегистрировать драйвер JDBC. Многие JAR-файлы прикладного интерфейса JDBC (например, драйвер базы данных Derby, входящий в состав версии Java SE 7) автоматически регистрируют класс драйвера. В этом случае вы можете пропустить этап ручной регистрации. JAR-файл может автоматически зарегистрировать класс драйвера, если он содержит файл META-INF/services/java.sql.Driver. Чтобы убедиться в этом, достаточно распаковать JAR-файл драйвера.

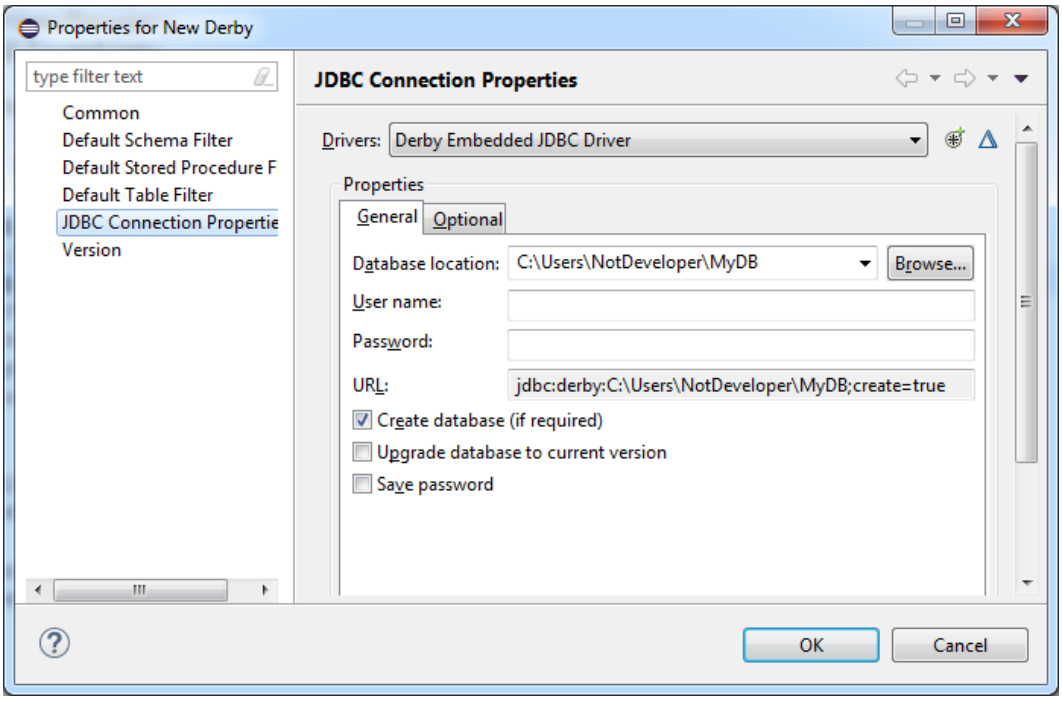


Рисунок 9.3 – Свойства соединения с базой данных.

Если же JAR-файл драйвера не поддерживает автоматическую регистрацию, вам придется выяснить имена классов драйверов JDBC, используемых поставщиком вашей базы данных. Типичными именами классов драйверов являются следующие:

org.apache.derby.jdbc.ClientDriver

org.apache.derby.jdbc.EmbeddedDriver

org.postgresql.Driver

Зарегистрировать драйвер с помощью класса DriverManager можно двумя способами. Один способ состоит в том, чтобы загрузить класс драйвера в

программу на Java, как показано в приведенной ниже строке кода, где выполняется статический инициализатор, который и осуществляет регистрацию загружаемого драйвера:

Class.forName("org.apache.derby.jdbc.EmbeddedDriver");

Другой способ состоит в том, чтобы задать свойство jdbc.drivers. Это свойство можно указать в качестве аргумента непосредственно в командной строке.

Рассмотрим пример кода Java-приложения, тестирующего соединение с базой данной, определенной на рис. 9.2. Путь к базе данных скопирован из окна на рис. 9.3.

```
1  package DerbyTest;
2  import java.sql.Connection;
3  import java.sql.DriverManager;
4  import java.sql.SQLException;
5
6  public class Starter {
7
8      public static void main(String[] args) {
9          try {
10             String drv = "org.apache.derby.jdbc.EmbeddedDriver";
11             Class.forName(drv);
12             String dbURL = "jdbc:derby:C:\\Users\\NotDeveloper\\MyDB;create=true";
13             Connection conn = DriverManager.getConnection(dbURL);
14             if (conn != null) {
15                 System.out.println("Connection successful!");
16                 conn.close();
17             }
18         }
19         catch (SQLException e){
20             e.printStackTrace();
21         }
22         catch(ClassNotFoundException e){
23             e.printStackTrace();
24         }
25     }
26 }
```

Рисунок 9.4 – Простейшая программа для тестирования соединения с базой данных.

На рис. 9.4 представлены следующие этапы создания подключения к базе данных:

1. Регистрация драйвера.

2. Задание строки подключения.

3. Создание объекта подключения типа Connection.

Диспетчер перебирает все зарегистрированные драйверы, пытаясь найти тот, который соответствует подчиненному протоколу, указанному в URL базы данных. Метод getConnection() возвращает объект типа Connection, который используется для выполнения команд SQL. Чтобы соединиться с базой данных, необходимо знать имя пользователя базы данных и пароль. В данном примере используется БД без пароля.

Реализация операций баз данных

На рис. 9.5 представлен код класса, реализующего две операции: подключение к базе данных и получение данных из таблицы «Goods».

```
 9 public class DBSelector {
10     public static Connection getConnection(){
11         Connection con = null;
12         try {
13             String drv = "org.apache.derby.jdbc.EmbeddedDriver";
14             Class.forName(drv);
15             String dbURL = "jdbc:derby:C:\\Users\\NotDeveloper\\MyDB;create=true";
16             con = DriverManager.getConnection(dbURL);
17             if (con != null) {
18                 System.out.println("Connection successful!");
19             }
20         }
21         catch (SQLException e){ e.printStackTrace(); }
22         catch(ClassNotFoundException e){ e.printStackTrace(); }
23         return con;
24     }
25 
26     public static void printData(){
27         Connection con = getConnection();
28         if(con != null){
29             Statement stat;
30             try {
31                 stat = con.createStatement();
32                 ResultSet res = stat.executeQuery("SELECT * FROM GOODS");
33                 while(res.next()){
34                     System.out.println(res.getString(1) + " | " +
35                             res.getString(2) + " | " +
36                             res.getString(3) + " | " +
37                             res.getString(4) + " | ");
38                 }
39                 con.close();
40             } catch (SQLException e) {
41                 e.printStackTrace();
42             }
43         }
44     }
45 }
```

Рисунок 9.5 – Получение данных из таблицы «Goods».

На рис. 9.6 представлен код метода main(). На рис. 9.7 приводится консольный вывод программы.

```
6  public class Starter {
7
8      public static void main(String[] args) {
9          DBSelector.printData();
10     }
11 }
```

Рисунок 9.6 – Метод main().

```
Connection successful!
1 | Хлеб | 30.0 | Продукты |
2 | Молоко | 21.0 | Продукты |
3 | Моющее средство | 125.0 | Химия |
4 | Рубашка | 1200.0 | Одежда |
5 | Туфли | 2500.0 | Одежда |
6 | Творог | 54.0 | Продукты |
```

Рисунок 9.7 – Вывод данных из БД.

В рамках представленного примера можно осуществлять ввод данных с использованием конструкции:

```
stat = con.createStatement();
stat.executeUpdate("INSERT INTO Goods VALUES (7, 'Соль', 10.50, 'Продукты')");
```

Рисунок 9.8 – Вставка записи с использованием executeUpdate().

Метод executeUpdate() возвращает количество строк, полученных из таблицы базы данных в результате выполнения команды SQL, или же нуль строк для команд, которые не возвращают количество строк из таблицы. Так, в результате приведенного выше вызова метода executeUpdate() возвращается 1.

Вызывая метод executeUpdate(), можно выполнять команды INSERT, UPDATE и DELETE, а также команды определения данных, в том числе CREATE TABLE и DROP TABLE. Но для выполнения команды SELECT необходимо вызвать другой метод, а именно executeQuery(). Имеется также универсальный метод execute(), с помощью которого можно выполнять

произвольные команды SQL, но он применяется в основном для составления запросов в диалоговом режиме.

Метод executeQuery() возвращает объект типа ResultSet, как показано на рис. 9.5. Этот объект можно использовать для построчного просмотра результатов выполнения запроса.

Строки располагаются в результирующем наборе в произвольном порядке. Если порядок их следования важен, его необходимо установить с помощью оператора ORDER BY. При обработке отдельной строки обычно требуется получить содержимое отдельных полей (или столбцов). Для этой цели имеется целый ряд методов доступа к полям (или столбцам).

Вв примере на рис. 9.5 используется метод getString() для получения строкового значения каждого поля.

Для каждого метода getString(), getDouble(), getInt() существует две версии: одна – со строковым параметром, вторая – с целочисленным. Если метод доступа вызывается с числовым параметром, данные извлекаются из столбца с указанным номером. Для каждого типа данных Java предусмотрен отдельный метод доступа. Нумерация столбцов в выборке начинается с 1. Если же метод доступа вызывается со строковым параметром, данные извлекаются из столбца с указанным именем.

Каждый из рассмотренных классов Connection, Statement и ResultSet содержит набор методов, которые используются при разработке приложений баз данных. Все методы рассмотрены в следующих таблицах.

В таблице 9.1 представлены описания методов класса Connection.

Таблица 9.1 – Методы класса Connection.

Метод	Описание
Statement createStatement()	Создает объект типа Statement, который может использоваться для выполнения команд SQL без параметров.
void close()	Немедленно разрывает текущее соединение и освобождает созданные для него ресурсы JDBC.

В таблице 9.2 приводятся основные методы класса Statement. Класс Statement представляет собой тип данных, ассоциированный с командой, посылаемой серверу БД.

Таблица 9.2 – Методы класса Statement.

Метод	Описание
ResultSet executeQuery(String sqlQuery)	Выполняет команду SQL из указанной символьной строки и возвращает объект типа ResultSet с результатами выполнения этой команды.
int executeUpdate(String sqlStatement)	Выполняет команды SQL типа insert, update и delete из указанной символьной строки, а также команды определения данных create table и drop table. Возвращает количество строк, обработанных при выполнении данной команды SQL, или нулевое значение, если для данной команды SQL не установлен подсчет обновлений.
boolean execute(String sqlStatement)	Выполняет команду SQL из указанной символьной строки и возвращает логическое значение true, если эта команда предоставляет результирующий набор, а иначе – логическое значение false. Может сформировать множество результирующих наборов и подсчетов обновлений. Для доступа к данным, полученным в результате выполнения данной команды SQL, следует вызвать метод getResultSet() или getUpdateCount().
ResultSet getResultSet()	Возвращает объект типа ResultSet с результатами выполнения предыдущей команды SQL или пустое значение null, если выполнение предыдущей команды не дало никаких результатов. Этот метод следует вызывать только один раз для каждой выполняемой команды SQL.
int getUpdateCount()	Возвращает количество строк, обработанных при выполнении предыдущей команды обновления, или значение -1, если для данной команды SQL не установлен подсчет обновлений. Этот метод следует вызывать только один раз для каждой выполняемой команды SQL.
void close()	Закрывает данную команду SQL и связанные с ней результирующие наборы.
boolean isClosed()	Возвращает логическое значение true при закрытии данной команды SQL.
void closeOnCompletion()	Закрывает данную команду SQL, как только будут закрыты все связанные с ней результирующие наборы.

В таблице 9.3 представлены основные возможности, предоставляемые программисту классом ResultSet. Данный класс ассоциируется с результирующим набором данных.

Таблица 9.3 – Методы класса ResultSet.

Метод	Описание
boolean next()	Перемещает указатель текущей строки в результирующем наборе на одну позицию вперед. После прохождения последней строки возвращает логическое значение false. Следует иметь в виду, что данный метод нужно непременно вызвать для перемещения указателя к первой строке в результирующем наборе.
int executeUpdate(String sqlStatement)	Выполняет команды SQL типа insert, update и delete из указанной символьной строки, а также команды определения данных create table и drop table. Возвращает количество строк, обработанных при выполнении данной команды SQL, или нулевое значение, если для данной команды SQL не установлен подсчет обновлений.
Xxx getXxx(int columnNumber)	Xxx обозначает тип данных, например int, double, String, Date и т.д. Возвращает значение столбца, задаваемого номером или меткой, с преобразованием в указанный тип данных. Следует иметь в виду, что допускаются не все варианты преобразования типов. Метка столбца – это метка, указываемая в операторе AS команды SQL, или имя столбца, если оператор AS не используется.
Xxx getXxx(String columnLabel)	
int findColumn(String columnName)	Возвращает номер столбца с указанным именем.
void close()	Немедленно закрывает текущий результирующий набор.
boolean isClosed()	Возвращает логическое значение true при закрытии данной команды.

Каждый объект типа Connection может создать один или несколько объектов типа Statement. Один и тот же объект типа Statement можно использовать для выполнения нескольких не связанных между собой команд и запросов. Но для такого объекта допускается наличие не более одного открытого результирующего набора. Если же требуется выполнить несколько

команд и одновременно проанализировать их результаты, то для этого понадобится несколько объектов типа Statement.

Метод close() класса Statement автоматически закрывает результирующий набор, связанный с объектом данного класса, если, конечно, этот набор открыт при выполнении соответствующей команды. Аналогично метод close() класса Connection закрывает все объекты данного класса, открытые для соединения с базой данных.

Применение прокручиваемых результирующих наборов

По умолчанию результирующие наборы не являются прокручиваемыми или обновляемыми. Для организации прокрутки результатов выполнения запроса необходимо получить объект типа Statement следующим образом:

Statement stat = conn.createStatement(type, concurrency);

Допустимые значения параметров type и concurrency перечислены в таблицах 9.4 и 9.5 соответственно.

Таблица 9.4 – Значения типа ResultSet.

Значение	Описание
TYPE_FORWARD_ONLY	Без прокрутки (по умолчанию).
TYPE_SCROLL_INSENSITIVE	С прокруткой, но без учета изменений в базе данных.
TYPE_SCROLL_SENSITIVE	С прокруткой и с учетом изменений в базе данных.

Таблица 9.5 – Параллельные значения ResultSet.

Значение	Описание
CONCUR_READ_ONLY	Без редактирования и обновления базы данных (по умолчанию).
CONCUR_UPDATABLE	С редактированием и обновлением базы данных.

Если этого не требуется делать результирующий набор прокручиваемым, необходимо использовать значение ResultSet.TYPE_FORWARD_ONLY.

Если требуется сделать результирующий набор прокручиваемым и он не должен отражать те данные, которые были изменены в базе данных после выполнения запроса, то устанавливается параметр ResultSet.TYPE_SCROLL_INSENSITIVE, то есть результирующий набор не «реагирует» на те изменения, которые произошли в базе данных после выполнения запроса.

После такого создания объекта Statement можно прокручивать все результирующие наборы, возвращаемые при вызовах метода executeQuery(). Получаемый в итоге результирующий набор содержит курсор, устанавливаемый на текущей позиции.

Прокрутка реализуется с использованием методов класса ResultSet. Для перехода к предыдущим записям в результирующем наборе служит метод previous(), который возвращает логическое значение true, если курсор находится на конкретной строке в результирующем наборе, и логическое значение false, если курсор находится перед первой строкой.

Для перемещения курсора на n строк вперед или назад вызывается метод relative(n). При положительных значениях параметра n курсор перемещается вперед, а при отрицательных – назад (нулевое значение параметра n не приводит ни к каким перемещениям). Если попытаться переместить курсор за пределы текущего ряда строк, он расположится за последней строкой или же перед первой строкой в зависимости от знака параметра n. После этого метод relative() возвращает логическое значение false, а перемещение курсора прекращается. Данный метод возвращает логическое значение true только в том случае, если курсор устанавливается на конкретной строке.

Курсор можно установить на конкретной строке под номером n, вызвав метод absolute(n), а получить текущий номер строки можно с использованием метода getRow().

Существуют также методы, осуществляющие позиционирование курсора в предопределенные позиции. В таблице 9.6 представлены методы класса ResultSet для работы с прокручиваемыми наборами. Использование этих

методов существенно облегчает работу программиста, но следует понимать, что подобный механизм требует больших затрат ресурсов ЭВМ по сравнению с использованием наборов без прокрутки.

Таблица 9.6 – Методы ResultSet, реализующие функции работы с прокручиваемым набором.

Метод	Описание
int getType()	Возвращает одну из констант TYPE_FORWARD_ONLY, CONCUR_UPDATABLE или TYPE_SCROLL_SENSITIVE, обозначающих тип результирующего набора.
int getConcurrency()	Возвращает константу CONCUR_READ_ONLY или CONCUR_UPDATABLE, обозначающую способ параллельного обращения к результирующему набору (только для чтения или для обновления).
boolean previous()	Перемещает курсор к предыдущей строке. Возвращает логическое значение true, если курсор устанавливается на строке, или логическое значение false, если он устанавливается перед первой строкой.
int getRow()	Получает номер текущей строки. Нумерация строк начинается с 1.
boolean absolute(int r)	Перемещает курсор к строке с номером r. Возвращает логическое значение true, если курсор устанавливается на строке.
boolean relative(int d)	Перемещает курсор на d строк. Если значение параметра d меньше нуля, то перемещение происходит в обратном направлении. Возвращает логическое значение true, если курсор устанавливается на строке.
boolean first()	Устанавливает курсор перед первой строкой.
boolean last()	Устанавливает курсор за последней строкой.
boolean isFirst()	Проверяет, находится ли курсор на первой строке.
boolean isLast()	Проверяет, находится ли курсор на последней строке.
boolean isBeforeFirst()	Проверяет, находится ли курсор перед первой строкой.
boolean isAfterLast()	Проверяет, находится ли курсор за последней строкой.
void moveToInsertRow()	Перемещает курсор на строку вставки. Строка вставки – это специальная строка, которая служит для вставки новых данных с помощью методов updateXxx() и innsertRow().

void moveToCurrentRow()	Перемещает курсор из строки вставки на строку, где он находился до вызова метода moveToInsertRow().
void insertRow()	Вводит содержимое строки вставки в базу данных и результирующий набор.
void deleteRow()	Удаляет текущую строку из базы данных и результирующего набора.
void updateXxx(int column, Xxx data)	Xxx обозначает тип данных, например int, double, String, Date и т.д. Обновляют содержимое указанного столбца текущей строки в результирующем наборе.
void updateXxx(String columnName, Xxx data)	
void updateRow()	Передает обновления текущей строки в базу данных.
void cancelRowUpdates()	Отменяет обновление текущей строки.

Применение обновляемых результирующих наборов

Если требуется отредактировать данные, полученные по запросу в результирующем наборе, а также автоматически обновить базу данных, такой набор нужно сделать обновляемым. Обновляемые результирующие наборы совсем не обязательно должны быть прокручиваемыми. Но если требуется предоставить пользователю возможность редактировать данные, то они должны допускать и прокрутку.

Для получения обновляемого результирующего набора служит приведенный код, представленный на рис. 9.9. Результирующие наборы, возвращаемые методом executeQuery(), становятся в итоге обновляемыми.

```
stat = con.createStatement(
        ResultSet.TYPE_SCROLL_INSENSITIVE,
        ResultSet.CONCUR_UPDATABLE);
```

Рисунок 9.9 – Создание обновляемого результирующего набора.

Методы, реализующие обновление результирующего набора рассмотрены в таблице 9.6. Например, добавление строки можно реализовать следующим кодом (рис. 9.10). Сравните данный метод добавления строки в базу данных и код, представленный на рис. 9.8.

```
33                  stat = con.createStatement(
34                          ResultSet.TYPE_SCROLL_INSENSITIVE,
35                          ResultSet.CONCUR_UPDATABLE);
36                  ResultSet res = stat.executeQuery("SELECT * FROM GOODS");
37                  res.moveToInsertRow();
38                  res.updateInt("ID", 8);
39                  res.updateString("Name", "Кофе");
40                  res.updateDouble("Price", 165.00);
41                  res.updateString("GrName", "Продукты");
42                  res.insertRow();
43                  res.moveToCurrentRow();
```

Рисунок 9.10 – Добавление строки через обновляемые наборы.

На данном этапе изучения технологий работы с базами данных, которые используются в Java, студент должен обладать достаточным теоретическим материалом для построения приложений Java для максимально эффективного использования реляционных хранилищ данных.

Методика выполнения

Создайте базу данных в соответствии с вариантом. Реализуйте экспорт данных из БД в текстовый файл, в формат табличного процессора.

Вариант	Задание

1	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: количество преподавателей; сумму зарплат преподавателей по каждой кафедре.

2	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: суммарную стоимость всех автомобилей; автомобиль с наименьшим пробегом.

3	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: количество автобусных рейсов; суммарную стоимость билетов рейсов самолетов; самый дорогой билет.

4	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждому курсу; количество студентов без задолженностей.

5	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную стоимость товаров по каждой группе; самый дорогой товар в каждой товарной группе.

6	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждой группе; склад, содержащий максимальное количество продукции.

7	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: для каждого строения рассчитать стоимость квадратного метра; суммарную стоимость объектов по каждому типу.

8	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: самую дешевую книгу в каждом жанре; среднюю стоимость книг в каждом жанре.

9	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: цену каждого товара за вычетом скидки; суммарную стоимость товаров по каждой группе.

10	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: среднюю цену товара по каждому магазину; суммарную стоимость для каждого товара.

11	Исходный файл содержит список преподавателей в формате «ФИО, должность, кафедра, заработная плата» (произвольное количество строк):
	`Иванов С.Е.| доцент | ИСТ | 56.00`
	`Петров А.А. | профессор | ИБАС | 100.23`
	`Новиков П.С. | ст. преподаватель | ИСТ | 35.00`
	`Ганотченко Е.П. | профессор | ИБАС | 200.45`
	`Петров А.А. | профессор | ИБАС | 105.67`
	Необходимо вычислить: суммарный фонд заработной платы; среднюю зарплату по каждой кафедре.

12	Исходный файл содержит список преподавателей в формате «Модель, производитель, стоимость, пробег» (произвольное количество строк):
	`ВАЗ-2107; Автоваз; 140000.57; 16000`
	`Гога-1; Хуань Вей Трейд; 50.00; 1000`
	`ВАЗ-5000; Автоваз; 1400000.00; 1000`
	`X35; Hyundai; 750000; 1`
	Необходимо вычислить: средний пробег по каждому производителю; среднюю стоимость автомобилей по каждому производителю.

13	Исходный файл содержит список преподавателей в формате «Транспорт; пункт отправки; пункт назначения; стоимость билета» (произвольное количество строк):
	`Самолет*Москва*Пекин*30000.00`
	`Автобус*Ставрополь*Москва*3000.00`
	`Автобус*Невинномысск*Ставрополь*10.00`
	`Автобус*Москва*Владивосток*50000`
	`Самолет*СПб*Москва*1500`
	Необходимо вычислить: среднюю стоимость билета по каждому виду транспорта; количество рейсов из Москвы.

14	Исходный файл содержит список преподавателей в формате «ФИО, группа, курс, количество задолженностей» (произвольное количество строк):
	`Иванов Е.А.*ИСТ-б-о-121*3*2`
	`Петров А.А.*ИСТ-б-о-111*4*1`
	`Кононов А.Е.*БАС-б-о-111*4*0`
	`Катченко Е.А.*БАС-б-о-121*3*1`
	Необходимо вычислить: суммарное количество задолженностей по каждой группе; среднее количество задолженностей по каждому курсу.

15	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена закупки, цена продажи» (произвольное количество строк):
	`Молоко, 0.5 л % Продукты % 10.00 % 23.00`
	`Хлеб белый % Продукты % 15.00 % 22.00`
	`Молоко, 1.0 л % Продукты % 18.00 % 30.00`
	`Туалетный утенок % Химия % 11.00 % 17.58`
	`Фейри, 0.5 л % Химия % 17.00 % 88.00`
	Необходимо вычислить: суммарную прибыль от продажи товаров; среднюю цену закупки по каждой группе.

16	Исходный файл содержит список преподавателей в формате «Товар, товарная группа, цена, склад» (произвольное количество строк):
	`Кабель сетевой % Сетевое оборудование % 10.00 % Пулково 1`
	`Коммутатор ASUS X100 % Сетевое оборудование % 15.00 % Склад №7`
	`ПО Windows 8.1, 1.0 л % ПО % 18.00 % Склад №19`
	`ПО Office 365 % ПО % 11.00 % Склад №19`
	`Коннектор 234511 % Сетевое оборудование % 17.00 % Склад №7`
	Необходимо вычислить: среднюю стоимость товаров по каждому складу; суммарную стоимость по каждой группе.

17	Исходный файл содержит список преподавателей в формате «Тип строения, количество комнат, метраж, стоимость» (произвольное количество строк):
	`Дом*6*150*3500000`
	`Квартира*1*49*790000`
	`Квартира*2*59*1500000`
	`Квартира*3*70*3000000`
	`Комната*1*12*500000`
	Необходимо вычислить: вывести объект с максимальной стоимостью квадратного метра; среднюю стоимость квадратного метра по объектам типа «Квартира».

18	Исходный файл содержит список преподавателей в формате «Название книги, жанр, цена» (произвольное количество строк):
	`Привет с того света | Детектив | 150.00`
	`Кто подставил кролика | Детектив | 200.10`
	`Почемучка | Детская литература | 60.22`
	`Капитошка | Детская литература | 230.14`
	`Звезда 112 | Фантастика | 12.90`
	Необходимо вычислить: суммарную стоимость книг по каждому жанру; среднюю стоимость книг по жанру «Фантастика».

19	Исходный файл содержит список преподавателей в формате «Товар, группа, стоимость, процент скидки» (произвольное количество строк):
	`Молоко \ Продукты \ 40.00 \ 1`
	`Тряпка \ Хозтовары \ 10.00 \ 5`
	`Порошок \ Хозтовары \ 750.00 \ 10`
	`Мясо \ Продукты \ 350 \ 5`
	`Швабра \ Хозтовары \ 900.00 \ 5`
	Необходимо вычислить: среднюю цену товара (с учетом скидки) по каждой товарной группе; товар с минимальной скидкой.

20	Исходный файл содержит список преподавателей в формате «Товар, цена, количество, магазин» (произвольное количество строк):
	`Рис 1 кг. / 120.11 / 54 / Магнит`
	`Ручка шариковая / 23.00 / 15 / ТЦ Карандаш`
	`Тетрадь 12 л. / 15.00 / 200 / ГеоКнига`
	`Тетрадь 60 л. / 90.70 / 300 / ГеоКнига`
	`Торт 1 кг. / 450 / 4 / Магнит`
	Необходимо вычислить: суммарную стоимость товаров в каждом магазине; магазин с минимальным товарным ассортиментом.

Контрольные вопросы

1. Перечислите и охарактеризуйте основные типы драйверов JDBC. Какие типы являются устаревшими?

2. Поясните назначение основных элементов схемы взаимодействия типа «клиент-сервер».

3. Поясните архитектуру n-уровневых приложений Java.

4. Что такое JDBC?

5. Что такое драйвер базы данных? Какими способами можно его зарегистрировать?

6. Какие средства визуального проектирования баз данных присутствуют в среде разработки Eclipse?

7. Опишите основные этапы создания подключения к базе данных через JDBC. Какими классами реализуется каждый этап?

8. Какими способами можно добавить строки в базу данных? Какие классы реализуют данные способы?

9. Поясните назначение основных методов класса Connection.

10. Поясните назначение основных методов класса Statement.

11. Поясните назначение основных методов класса ResultSet.

12. Что такое «прокручиваемые результирующие наборы»? Чем они отличаются от обычнх наборов данных?

13. Каково назначение обновляемых результирующих наборов? Какие методы доступны программисту при работе с данных типом результирующих наборов?

Литература

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следующие источники: [6-8].

ЗАКЛЮЧЕНИЕ

В учебном пособии представлены методические указания к лабораторным работам, которые необходимы студентам при разработке сложных приложений, основанных на объектных технологиях. Учебное пособие (лабораторный практикум) по дисциплине «Технологии суперкомпьютерных кластеров и распределенных систем» предоставляет студентам материал для освоения технологий баз данных, сетевого программирования, обработки исключений и проектирования приложений с использованием шаблонов.

Материал лабораторных работ представлен на базе современной среды разработки программных средств – интегрированной среды разработки Eclipse.

В качестве основы построения лабораторного практикума предложена технология Java SE. Данная технология предлагает широкий спектр алгоритмов и технологий разработки программного обеспечения: широкий набор библиотечных типов; специализированные синтаксические конструкции; современные примитивы программирования; компоненты разработки графических приложений; классы для работы с объектами файловой системы. Все представленные технологии в рамках одного языка позволяют студентам получить практический опыт использования современных инструментов разработки объектно-ориентированных приложений.

СПИСОК ЛИТЕРАТУРЫ

Список основной литературы

1. Шилдт, Г. Java. Полное руководство. 8-е изд. / Г. Шилдт; пер. с. англ. В.А. Коваленко. – М.: ООО «И.Д. Вильямс», 2012. – 1104 с.

2. Корнелл, Г. Java. Библиотека профессионала, том 1. Основы. 9-е изд. / Г. Корнелл, К.С. Хорстманн; пер. с. англ. И.В. Берштейн. – М.: ООО «И.Д. Вильямс», 20110. – 864 с.

3. Корнелл, Г. Java. Библиотека профессионала, том 2. Расширенные средства. 9-е изд. / Г. Корнелл, К.С. Хорстманн; пер. с. англ. И.В. Берштейн. – М.: ООО «И.Д. Вильямс», 20110. – 864 с.

4 Язык программирования C# 5.0 и платформа .NET 10.5. 6-е изд. Пер. с англ. / Э. Троелсен. – М.:Изд. «Издательский дом Вильямс». 2013. – 1312 с.

5. Албахари Дж. C# 5.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – М.:Изд. «Издательский дом Вильямс». 2013. – 1008 с.

Список дополнительной литературы

6. Стиллмен Э. Изучаем C#. 3-е изд. / Э. Стиллмен, Дж. Грин. – СПб.: Питер. 20110. – 816 с.

7. Флентов, М. Библия C#. 2-е изд. / М. Флентов. – СПб.: БХВ-Петербург, 2011. – 560 с.

8. Ватсон, Б. C# 10.0 на примерах: пер. с англ. / Б. Ватсон. – СПб.: БХВ-Петербург, 2011. – 608 с.

15. Буч, Г. Объектно-ориентированный анализ и проектирование с примерами приложений, 3-е изд.: Пер. с англ. – М.: 000 «И.Д. Вильяме», 2008. – 720 с.: ил. – Парал. тит. англ.

10. Петцольд Ч. Программирование для Microsoft Windows 8. 6-е изд. / Ч. Петцольд. – СПб.: Питер. 2013 – 1008 с.

11. Гранд, М. Шаблоны проектирования в Java / М. Гранд; пер. с. англ. С. Беликовой. – М.: Новое знание, 20010. – 559 с.

12. Монахов, В. Язык программирования Java и среда NetBeans. 3-е изд., перераб. и доп. / В.В. Монахова. – СПб.: БХВ-Петербург, 2011. – 704 с.

13. Блох, Дж. Java. Эффективное программирование. 2-е изд. / Дж. Блох; пер. с. англ. Е. Коротылева. – М.: Лори, 20110. – 461 с.

110. Блинов, И. Java: методы программирования: уч.-мет. Пособие / И.И. Блинов, В.С. Романчик. – Минск: «Четыре четверти», 2013. – 896 с.

Технологии суперкомпьютерных кластеров и распределенных систем

УЧЕБНОЕ ПОСОБИЕ

(лабораторный практикум)

для студентов специальности

09.04.02 «Информационные системы и технологии»

Автор: Е.И. Николаев

Редактор:

Подписано в печать _____г.

Формат 60х84 1/16 Усл.п.л. – 4,25. Уч.-изд.л. – 3,10.

Бумага газетная. Печать офсетная. Заказ № ___ Тираж ___ экземпляров

ФГАОУ ВПО «Северо-Кавказский федеральный университет»

355000, г. Ставрополь, ул. Пушкина, 1

Издательство СКФУ