МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ»

УТВЕРЖДАЮ

Зав. кафедрой АСУ, профессор

 А.М. Кориков

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Методические рекомендации по практическим занятиям

Учебно-методическое пособие

для студентов уровеня основной образовательной программы магистратура
направления подготовки 010400.68 «Прикладная математика и информатика»
профиля Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей

Разработчик

доцент кафедры АСУ

В.Г. Резник

2012

Резник В.Г.

Современные компьютерные технологии. Методические рекомендации по практическим занятиям: Учебно-методическое пособие. – Томск, ТУСУР, 2012. – 36 с.

Данное учебно-методическое пособие предназначено для выполнения практических занятий по дисциплине «Современные компьютерные технологии» для студентов уровня основной образовательной программы магистратура направления подготовки 010400.68 «Прикладная математика и информатика» профиля «Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей».

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Рассматриваемое учебное пособие содержит методические рекомендации, обеспечивающие проведение практических занятий по дисциплине «Современные компьютерные технологии» (СКТ).

Все работы по практическим занятиям, в рамках дисциплины СКТ, проводятся на базе «Учебного программного комплекса кафедры АСУ» (УПК АСУ), созданного на основе операционной системы (ОС) Linux дистрибутива Xubuntu.

Данный УПК входит в состав общего учебного комплекса кафедры АСУ и включает распределенный доступ к общим файловым ресурсам кафедры.

Последовательность и изложение материала данного методического пособия предполагает, что студент:

- успешно успешно изучает теоретический материал по дисциплине «Современные компьютерные технологии»;
- владеет практическим умением, полученным в процессе выполнения лабораторных работ по данной дисциплине;
- успешно освоил навыки разработки прикладного ПО на языках программирования Java и SQL в среде разработки Eclipse EE;
- проводит предварительную самостоятельную работу, в пределах 2-х часов, для подготовки по каждому практическому занятию;
- завершает практические занятия оформлением письменного отчета.

Данное методическое пособие содержит шесть разделов, охватывающих все вопросы практических занятий. Названия разделов, перечисленные в оглавлении, соответствуют тематике проводимых занятий. В конце каждого раздела приведен список рекомендованных источников информации, необходимых для изучения теоретического материала и проведения практических работ по данной дисциплине.

Дополнительно, когда это необходимо, преподаватель предоставляет обучающемуся всю необходимую информацию в файлах формата pdf, а также требуемое программное обеспечение, в форматах предусмотренных заданием.

B.1 Состав ПО УПК АСУ для проведения практических занятий

Для проведения практических занятий используются инструментальные и функциональные возможности УПК АСУ. Минимальный необходимый набор дистрибутивов ПО для этих работ представлен в табл. B.1.

Общая директория нахождения всех дистрибутивов определена как: **/cdrom/casper/asu11upk/opt/**

Таблица В.1. - Перечень дистрибутивов ПО УПК АСУ

Файл	**Назначение дистрибутива**
libreOffice3.4.squashfs	Дистрибутив офисного приложения, содержащий инструменты для подготовки отчетов и пояснительной записки проекта.
java7.squashfs	Дистрибутив Java, включающий JDK, JRE и СУБД Apache Derby.
chromium.squashfs	Типовой браузер для отображения интерфейсной части проекта и просмотра исходного кода html-страниц.
eclipseED.squashfs	Дистрибутив инструментальной среды разработки Eclipse EE, включающий плагины для работы с Apache Derby.
apache-tomcat-7.0.30.zip	Архив Apache Tomcat, требующий развертывание в домашней директории пользователя (проектировщика): $HOME/tomcat

Необходимый для запуска перечень скриптов, предназначенный для запуска приложений, представлен в табл. В.2. Скрипты должны находиться в директории: **$HOME/bin/**

Таблица В.2. - Скрипты ПО УПК АСУ

Файл	**Назначение**
startOffice	Монтирует дистрибутив LibreOffice и запускает его.
startJava	Монтирует дистрибутив Java.
startChromium	Монтирует дистрибутив браузера chromium и запускает его.
startED	Монтирует дистрибутив Eclipse EE, восстанавливает из архива рабочую область проектирования $HOME/workspaceED и запускает среду разработки.

Для настройки личной рабочей среды пользователя, необходимо использовать файлы:

- **/etc/host** — для именования URL ресурсов распределенного приложения;
- **$HOME/.bashrc** — для задания переменных среды, необходимых для правильного функционирования проектируемого приложения.

Для типового размещения ПО и данных занятия, в домашней директории пользователя, используются следующие директории:

- **workspaceED** — для рабочей области проектов инструментальной среды Eclipse EE;
- **tomcat** — для ПО и сервлетов Apache Tomcat;
- **databases** — для локальных баз данных СУБД Apache Derby.

B.2 Перечень информационных объектов для практических занятий

Прикладная тематика каждого практического занятия предполагает наличие информационных объектов, которые служат исходными данными для отдельных задач, решаемых обучающимися.

Общее требование к современным компьютерным технологиям — это интеграция в *распределенные системы обработки данных* (РСОД).

Обобщенным примером распределенных систем обработки данных, соответствующим направленности обучения студентов кафедры АСУ, являются *автоматизированные информационные системы* (АИС). Такие системы входят в состав АСУ любого предприятия.

Характерные черты АИС, как РСОД:

- наличие множества, достаточно автономных, центров обработки и хранения информации или *сосредоточенных СОД* (ССОД);
- возможность удаленного интерактивного взаимодействия с каждой такой ССОД;
- необходимость информационного обмена между ССОД для обеспечения сервиса интерактивного взаимодействия с системой.

Конкретизируя РСОД, как АИС некоторого предприятия, можно считать, что предприятие условно состоит из ***набора департаментов*** (подразделений). Каждый департамент имеет собственный локальный сервер, содержащий:

- *базу данных* СУБД Apache Derby, хранящую информацию о деятельности предприятия в пределах компетентных полномочий департамента;
- *контейнер сервлетов* Apache Tomcat, обеспечивающий функциональные возможности департамента по доступу к базе данных с любого рабочего места распределенной системы посредством типового браузера Интернет.

Для учебных целей, перечисленные выше информационные объекты, могут быть построены на примере вуза:

- различные подразделения вуза можно рассматривать как отдельные департаменты предприятия;
- упрощенная информационная модель отдельного подразделения вуза может рассматриваться как исходные данные для создания ССОД.

Обучающиеся могут конкретизировать свои модели, опираясь на известные им подразделения университета — «ТУСУР».

Преподаватель, начиная практические занятия по данной дисциплине, выделяет каждому студенту отдельный информационный объект, ориентируясь на список тем, перечисленных в табл. В.3.

Таблица В.3. - Темы для создания индивидуальных информационных объектов

№	Код темы	Название темы
1	dep_ok	Департамент студенческого отдела кадров
2	dep_prep	Департамент преподавательского состава
3	dep_disc	Департамент учебного отдела
4	dep_decan	Департамент деканата
5	dep_kaf	Департамент кафедры (свободная тема)
6	dep_bibl	Департамент библиотеки (свободная тема)
7	dep_hoz	Департамент хозяйственной части (свободная тема)

1 ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ В СРЕДЕ СУБД DERBY

Цель данного практического занятия — создать упрощенную информационную модель департамента, которая будет рассматриваться как исходные данные для ССОД.

Построенная модель будет использоваться обучающимся во всех последующих практических занятиях по данному курсу.

Тематика модели выдается преподавателем каждому обучающемуся индивидуально.

Ориентировочный перечень тем содержится в табл. В.3.

1.1 Задания на построение информационных объектов

Функциональные возможности информационной модели и программного обеспечения каждого департамента должны обеспечивать:

- возможность редактирования информации, содержащейся в базе данных департамента, - администратором ССОД департамента ;
- доступ к информации департамента, без возможности редактирования, всеми участниками РСОД.

Минимальный состав информационной части моделей для различных тем приведен в табл. 1.1.

Модели обозначены как имена таблиц реляционной СУБД Derby.

Таблица 1.1. - Информационная часть баз данных системы

Код темы	**Таблица БД**	**Информационное содержание**
dep_ok	facultets students	- список факультетов вуза; - список студентов факультета.
dep_prep	kafedres prepods	- список кафедр факультета; - список преподавателей кафедры.
dep_disc	disciplines teachers	- список дисциплин, преподаваемых в вузе; - учителя кафедры, преподающие дисциплину.
dep_decan	groups lists_groups	- список групп факультета по кафедрам; - список студентов по группам.

Учитывая, что обучающиеся работают одной группой, необходимо придерживаться некоторой стандартизации имен объектов. Рекомендуется при обозначении баз данных придерживаться правилам формирования имен, как показано в табл. 1.2.

Таблица 1.2. - Базы данных департаментов вуза

Код темы	**База данных**	**Департамент вуза**
dep_ok	db_dep_ok	Департамент студенческого отдела кадров
dep_prep	db_dep_prep	Департамент преподавательского состава
dep_disc	db_dep_disc	Департамент учебного отдела
dep_decan	db_dep_decan	Департамент деканата
dep_kaf	db_dep_kaf	Департамент кафедры (свободная тема)
dep_bibl	db_dep_bibl	Департамент библиотеки (свободная тема)
dep_hoz	db_dep_hoz	Департамент хозяйственной части (свободная тема)

1.2 Технология создания информационных объектов

После выбора тематики моделей информационных объектов и уточнения их содержательной части с учетом данных таблиц 1.1 и 1.2, необходимо:

- развернуть на ЭВМ студента инструментальные среды Eclipse EE и СУБД Derby;
- подготовить сетевую среду исполнения приложений для создаваемого распределенного приложения;
- провести создание и сохранение информационных объектов в базах данных СУБД Derby.

Основная цель этой части практического занятия — формализация и согласование информационного содержания каждой части распределенного приложения.

Для достижения этой цели проводятся:

- согласование и настройка сетевой среды каждой операционной системы с учетом локальных особенностей ССОД и общими требованиями РСОД;
- определение и формализация таблиц баз данных прикладной части каждой локальной ССОД.

Настройка сетевой среды ССОД требует, чтобы сервер любого локального приложения имел:

- **IP-адрес**, по которому он доступен в сети;
- **номер порта** СУБД Apache Derby (**1527**) для удаленного доступа к ресурсам баз данных;
- **имя базы данных** всех приложений, определенные в таблице 5;
- **имена таблиц** всех приложений, определенные в таблице 4;
- **номер порта** Apache Tomcat (**8080**) для доступа к функциональности сервлетов;

- **имя проекта** (**deport**), реализующего каждую локальную часть приложения;
- **имена сервлетов** Apache Tomcat, которые обслуживают сервисные запросы от браузеров клиентов.

Для организации совместной работы всех компьютеров будущей РСОД, следует согласовать и настроить именование входящих в нее приложений.

Приемлемым вариантом для учебных целей является использование **алиесов** (сетевых имен), отображаемых в файле **/etc/hosts**.

Пример такой настройки приведен в листинге 1.1.

Листинг 1.1. - Файл /etc/hosts

```
#---------------------------------------
127.0.0.1 localhost
127.0.1.1 vgr #имя пользователя
#---------------------------------------
# IP-адрес   Алиесы имен приложений
#---------------------------------------
192.168.0.09 db_dep_ok    dep_ok    anna
192.168.0.10 db_dep_prep  dep_prep  ivan
192.168.0.11 db_dep_disc  dep_disc  sasha
192.168.0.12 db_dep_decan dep_decan ksenia
#---------------------------------------
```

Для формализации информационных частей РСОД следует разработать информационную структуру таблиц, сохраняемых в базе данных СУБД.

Структура таблиц должна полностью удовлетворять требованиям всего РСОД.

Дополнительно к таблицам, следует добавить ***общедоступный вариант доступа к информации***, который обычно выполняется в виде *информационных именованных представлений* **view**.

После согласования структур баз данных, требуется формализованное описание на языке SQL.

Пример такого описания, для таблиц **facultets** и **students**, а также их представлений **v_facultets** и **v_students**, приведен в листинге 1.2. Следует обратить внимание, что представления на языке SQL привязаны к схеме **APP**, которая является общедоступной для всех пользователей СУБД Derby.

Завершив создание информационных объектов ССОД, следует провести тестирование доступа к этим объектам.

Пример скрипта, обеспечивающего тестирование публичного доступа к базе данных ***db_dep_ok***, представлен в листинге 1.3.

Листинг 1.2. - Формализованное описание структур баз данных

```
------------------------------------------------------------
-- Скрипт создания таблицы facultets и students
-- БД: db_dep_ok
------------------------------------------------------------
-- Reznik, 19.10.2012
------------------------------------------------------------
-- Вариант: создавать БД
connect 'jdbc:derby://myhost:1527/db_dep_ok;
create=true;user=userdep;password=userdep;';
------------------------------------------------------------
drop view app.v_facultets;
drop view app.v_students;
drop table facultets;
drop table students;
------------------------------------------------------------
-- создание таблицы facultets

create table facultets (
      id int not null GENERATED ALWAYS AS identity (START WITH 1, INCREMENT BY
1),   -- номер факультета - ключ таблицы
      date_mod date default CURRENT_DATE,       -- дата модификации записи
      shortname varchar(40) default 'Новый',    -- имя факультета (короткое)
      fullname  varchar(256) default 'Новый факультет (до 255 зн.)',   -- полное
имя
      resume varchar(4096) default 'Комментарий (до 4096 зн.)',        --
комментарий
      primary key (id)
);
------------------------------------------------------------
-- создание некоторого количества записей таблицы facultets

insert into facultets(shortname,fullname) values('РТФ', 'Радио-технический
факультет');
insert into facultets(shortname,fullname) values('ФСУ', 'Факультет систем
управления');
------------------------------------------------------------
-- создание таблицы students

create table students (
      id int not null GENERATED ALWAYS AS identity (START WITH 1, INCREMENT BY
1),   -- номер студента - ключ таблицы
      idf int default 0, -- номер факультета: 0 - не распределен на факультет
      date_mod date default CURRENT_DATE,       -- дата модификации записи
      family varchar(40) default 'Новый',       -- фамилия
      name varchar(40) default 'до 40 зн.',     -- имя
      fathername  varchar(40) default 'до 40 зн.',   -- отчество
      resume varchar(4096) default 'Комментарий (до 4096 зн.)',  -- комментарий
      primary key (id)
);

------------------------------------------------------------
-- создание некоторого количества записей таблицы students

insert into students(family,name) values('Грибоедов', 'Александр');
insert into students(family,name) values('Квитко',    'Иван');
insert into students(family,name) values('Курьянович','Ксения');
insert into students(family,name) values('Тунгусова', 'Анна');

------------------------------------------------------------
```

```
-- создание общедоступной части БД

create view app.v_facultets as
      select * from userdep.facultets;

create view app.v_students as
      select * from userdep.students;

commit;

------------------------------------------------------------
-- проверка чтения из общедоступной части БД

commit;
select * from app.v_facultets;
select * from app.v_students;

------------------------------------------------------------
-- выход из БД

disconnect;
exit;
```

Листинг 1.3. - Пример скрипта для тестирования доступа к базе данных

```
------------------------------------------------------------
-- Скрипт тестирования таблиц v_facultets и v_students
-- БД: db_dep_ok
------------------------------------------------------------
-- Reznik, 19.10.2012
------------------------------------------------------------
-- Вариант: не создавать БД
connect 'jdbc:derby://dep_ok:1527/db_dep_ok;';
------------------------------------------------------------
-- В таком варианте доступ осуществляется к схеме APP БД
------------------------------------------------------------
-- выход из БД
commit;
select * from v_facultets;
select * from v_students;
disconnect;
exit;
------------------------------------------------------------
```

2 КОМПОНЕНТНОЕ РАЗВИТИЕ IDE ECLIPSE EE

2.1 Создание проекта расширения в среде Eclipse EE

В разработке.

2.2 Пример реализации модуля расширения Eclipse EE

В разработке.

3 РАСШИРЕНИЕ ВОЗМОЖНОСТЕЙ ПАКЕТА LIBREOFFICE

3.1 Сетевые объекты пакета LibreOffice

В разработке.

3.2 Пример удаленного доступа к пакету LibreOffice

В разработке.

4 МОДЕЛИРОВАНИЕ ПРОМЫШЛЕННОЙ ШИНЫ ESB

4.1 Сетевые объекты ESB

В разработке.

4.2 Пример реализации модели ESB

В разработке.

5 ПРОЕКТИРОВАНИЕ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ СИСТЕМ ОБЪЕКТОВ

5.1 Описание интерфейсной части ССОД

В разработке.

5.2 Реализация интерфейсной части ССОД

В разработке.

6 ПРОЕКТИРОВАНИЕ СЕРВИСНОГО ОБСЛУЖИВАНИЯ ОБЪЕКТОВ

Цель данного практического занятия — создание диалоговых страниц, обеспечивающих удаленное интерактивное взаимодействие пользователей с ССОД.

Данное взаимодействие называется ***трехуровневой схемой доступа*** к информационным объектам распределенной системы.

Поскольку современные распределенные системы очень часто для интерактивного для взаимодействия используют www-сервера или сервера приложений на основе протокола http, то, общепринято, говорить о ***доступе к сервисам***. Соответственно, схема доступа к информационным объектам, при которой сервера обеспечивают доступ к сервисам, а приложением клиента является браузер, стали называть ***сервисным обслуживанием***.

На практическом занятии, для организации сервисного обслуживания РСОД, используется ***контейнер сервлетов*** Apache Tomcat. В качестве браузера возможно использование любого современного представителя этого клиентского приложения.

Применительно к темам индивидуальных информационных моделей, обучающиеся решают ***следующие задачи***:

- *определение состава* интерфейсной и функциональной частей сервера ССОД, выраженной в перечислении jsp-страниц, необходимых для реализации интерфейсов пользователя, и перечислении поименного состава сервлетов, обеспечивающих функциональную поддержку интерфейсов;
- *определение средств* обработки ошибок, возникающих при отсутствии или неправильном задании параметров среды приложения
- реализация состава интерфейсной и функциональной частей ССОД в среде сервера (контейнера сервлетов) Apache Tomcat, с использованием инструментальной среды Eclipse EE.

6.1 Определение состава интерфейсной и функциональной частей ССОД

Определение состава интерфейсной и функциональной частей ССОД приведем на примере темы: **dep_ok** - «Департамент студенческого отдела кадров» (см. таблицы В.1, 1.1 — 1.3).

Согласно указанной темы, ***ССОД должна***: обеспечить модификацию списка факультетов и списка студентов, распределяемых по факультетам.

Для решения поставленной задачи достаточно наличие двух сервлетов и двух jsp-страниц, приведенных в табл. 6.1.

Таблица 6.1. - Состав интерфейсной и функциональной частей проекта

№	Объект проекта	Назначение
1	FalultetsOrg	Сервлет, обеспечивающий функциональную часть модификации таблицы **facultets**.
2	facultets.jsp	jsp-страница, обеспечивающая интерфейсную часть проекта для работы со списком факультетов.
3	StudentsOrg	Сервлет, обеспечивающий функциональную часть модификации таблицы **students**.
4	studentd.jsp	jsp-страница, обеспечивающая интерфейсную часть проекта для работы со списком студентов.

Указанные в табл. 6.1, информационные и программные объекты создаются в среде разработки Eclipse EE проекта **deport**.

На листинге 6.1 показан первоначальный шаблон сервлета **StudentsOrg**, вызывающий jsp-страницу **students.jsp**.

Листинг 6.1. - Первоначальный шаблон сервлета StudentsOrg

```
package ru.tusur.department;

import java.io.IOException;

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 StudentsOrg
 */
@WebServlet({ "/StudentsOrg", "/" })

public class StudentsOrg extends HttpServlet {
      private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public StudentsOrg() {
        super();
        // TODO Auto-generated constructor stub
    }

      protected void doGet(HttpServletRequest request,
```

```
                HttpServletResponse response)
                        throws ServletException, IOException {
            doPost(request, response);
        }

        protected void doPost(HttpServletRequest request,
                HttpServletResponse response)
                        throws ServletException, IOException {
            RequestDispatcher disp =
                        request.getRequestDispatcher("/WEB-INF/students.jsp");
            disp.forward(request, response);
        }
}
```

После создания шаблонов сервлетов создаются шаблоны jsp-страниц, которые должны иметь ссылки друг на друга для перехода с одной страницы на другую.

На листинге 6.2 приведен шаблон jsp-страницы для работы со списком студентов. Соответственно, на рис. 6.1 приведено отображение этой страницы в браузере пользователя.

Листинг 6.2. - Первоначальный шаблон jsp-страницы students.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Студенты ТУСУР</title>
</head>
<body LANG="ru-RU" BGCOLOR="#418fa2">
<h1 align="center">ТУСУР</h1><hr>
<h2 align="center">Студенты университета</h2>
<hr>
<%--
Меню-ссылка
--%>
<P ALIGN=CENTER >
<a HREF="FacultetsOrg"><b>Перейти на редактирование факультетов...</b></a>
</P>
<hr>

</body>
</html>
```

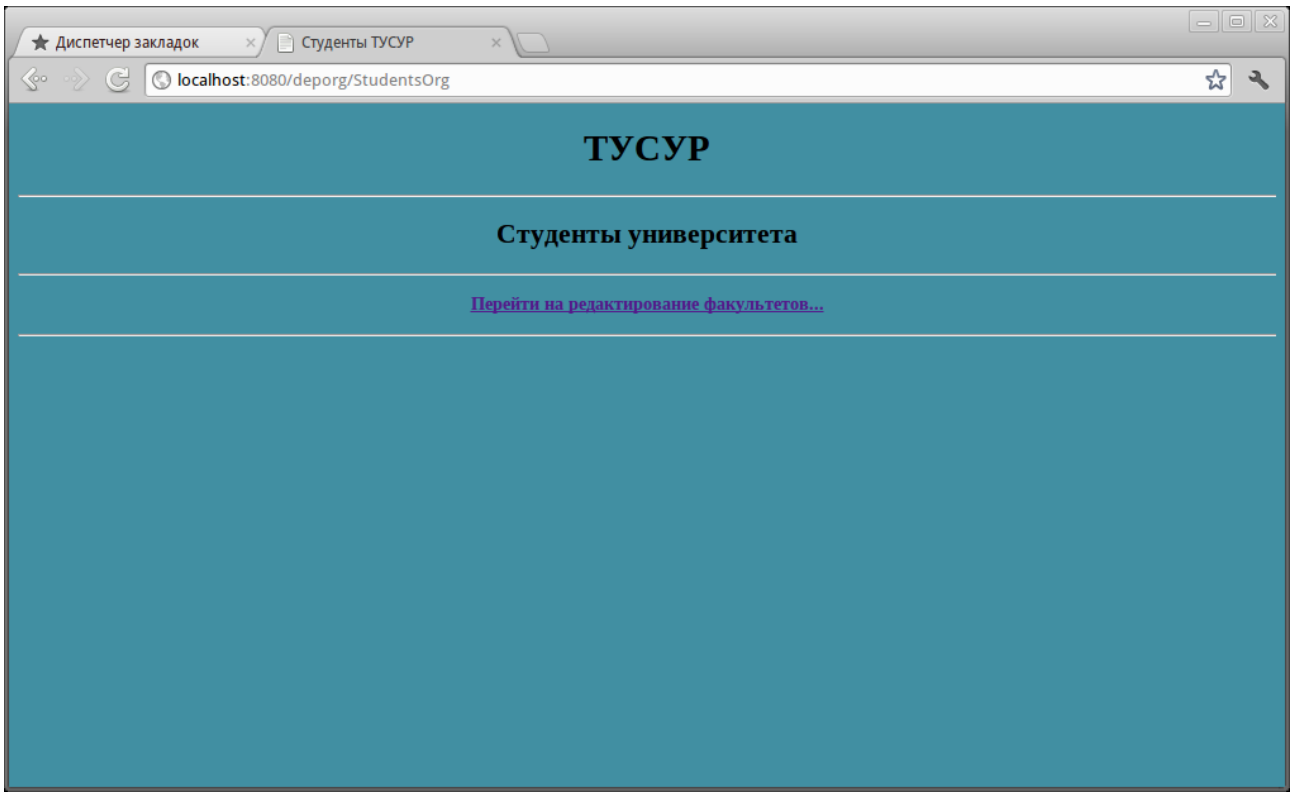

Рис. 6.1. Отображение шаблона jsp-страницы students.jsp

Поскольку каждое локальное приложение использует ***набор переменных среды***, необходимых для определения местоположения ресурсов приложения, то необходимо читать эти переменные при каждом старте сервлета.

Обычно, чтение всех внешних параметров сервлета включается в конструктор класса. Если, по каким-либо причинам, нужные переменные среды отсутствуют, то необходимо сообщить о такой ситуации для последующего ее устранения. Необходимый программный код обработки таких ситуаций нужно разработать до реализации остальной функциональной части проекта. Соответственно:

- листинг 6.3 показывает решение этого вопроса для сервлета **FacultetsOrg**.
- листинг 6.4 показывает jsp-страницу, отображающую обнаруженные ошибки.

Листинг 6.3 - Чтение и контроль наличия параметров среды сервлета FacultetsOrg

```
package ru.tusur.department;

import java.io.*;
import java.net.URLDecoder;
//import java.net.URLEncoder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
```

```
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;

@WebServlet("/FacultetsOrg")

//ОСНОВНОЙ КЛАСС СЕРВЛЕТА
public class FacultetsOrg extends HttpServlet {
      private static final long serialVersionUID = 1L;

      //ОБЩИЕ ПАРАМЕТРЫ (.bashrc, /etc/host)
      /*
export URL_OK="jdbc:derby://dep_ok:1527/db_dep_ok;"
export URL_PREP="jdbc:derby://dep_prep:1527/db_dep_prep;"
export URL_DISC="jdbc:derby://dep_disc:1527/db_dep_disc;"
export URL_DECAN="jdbc:derby://dep_decan:1527/db_dep_decan;"
export ADD_ADM="user=userdep;password=userdep;"
       */
      private String URL_OK     = null;
      private String URL_PREP   = null;
      private String URL_DISC   = null;
      private String URL_DECAN  = null;
      private String ADD_ADM    = null;

      private String  URL_ADM      = null;
      private boolean isParameters = true;

    public FacultetsOrg() {
        super();
        //Читаем переменные среды
        URL_OK    = System.getenv("URL_OK");
        URL_PREP  = System.getenv("URL_PREP");
        URL_DISC  = System.getenv("URL_DISC");
        URL_DECAN = System.getenv("URL_DECAN");
        ADD_ADM   = System.getenv("ADD_ADM");

        if (URL_OK != null && ADD_ADM != null)
            URL_ADM = URL_OK + ADD_ADM;
    }

    //БАЗОВЫЕ МЕТОДЫ doGet() и doPost()
      protected void doGet(HttpServletRequest request, HttpServletResponse
response)
                  throws ServletException, IOException {

            doPost(request, response);
      }

      protected void doPost(HttpServletRequest request, HttpServletResponse
response)
                  throws ServletException, IOException {

            System.out.println("doPost(): URL_ADM = " + URL_ADM);
            //Проверяю параметры
            if (!testParameters(request)){
                  //Вызов jsp-страницы с сообщением об ошибке
                  RequestDispatcher disp =
                  request.getRequestDispatcher("/WEB-INF/errParameters.jsp");
```

```
                disp.forward(request, response);
                return;
            }


            //Вызов jsp-страницы
            RequestDispatcher disp =
                        request.getRequestDispatcher("/WEB-INF/facultets.jsp");
            disp.forward(request, response);
        }

        //Тест параметров среды
        protected boolean testParameters(HttpServletRequest request)
                throws ServletException, IOException{
            String er = "";
            if (URL_OK == null){
                isParameters = false;
                er = "URL_OK...\n";
            }
            if (URL_PREP == null){
                isParameters = false;
                if (er.length() > 0) er += "#";
                er += "URL_PREP...\n";
            }
            if (URL_DISC == null){
                isParameters = false;
                if (er.length() > 0) er += "#";
                er += "URL_DISC...\n";
            }
            if (URL_DECAN == null){
                isParameters = false;
                if (er.length() > 0) er += "#";
                er += "URL_DECAN...\n";
            }
            if (ADD_ADM == null){
                isParameters = false;
                if (er.length() > 0) er += "#";
                er += "ADD_ADM...\n";
            }
            if (!isParameters) request.setAttribute("errParameters", er);
            return isParameters;
        }
}
```

Листинг 6.4 - Текст страницы errParameters.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h1>Установи переменные среды!</h1>
<hr>
<ul>
```

```
<%
String par = (String)request.getAttribute("errParameters");
if (par == null){
      out.println("<LI>Здесь - тоже ошибся!...</LI>");
}else{
      String [] aa = par.split("#");
      for (int i = 0; i < aa.length; i++)
            out.println("<LI>" + aa[i] +"</LI>");
}
%>
</ul>
<hr>
</body>
</html>
```

Полная реализация контрольного примера проекта проходит три стадии:

1. Подготовка интерфейсной части проекта, реализуемый на языке html для jsp-страниц;
2. Проектирование запросов браузера к серверу Apache Tomcat;
3. Последовательная реализация функциональной части проекта на языках JavaScript и java.

6.2 Подготовка интерфейсной части ССОД

Подготовка интерфейсной части ССОД состоит в программировании jsp-страниц на языке html с целью размещения на странице необходимых элементов управления: списков, полей ввода, кнопок и ссылок.

На этой стадии разработки функциональность страниц полностью отсутствует.

Главная задача — определить необходимые элементы интерфейса, его удобное расположение и решить минимальные вопросы дизайна приложения.

Пример такого интерфейса для работы со списком факультетов приведен на рис. 6.2.

Далее, переходим к организации диалога между браузером клиента и сервером ССОД.

Диалог между браузером и www-сервером возникает, когда браузер, по каким-либо причинам посылает запрос серверу.

Чтобы запрос браузера содержал «смысл», в html-страницы вводятся ***формы с именованными полями***.

Имена и значения этих полей передаются серверу по методу GET или POST после нажатия клавиши типа «submit».

Графический пример такой формы и показан на рис. 6.2.

Соответственно, в листинге 6.5, приведен пример этой формы на языке html.

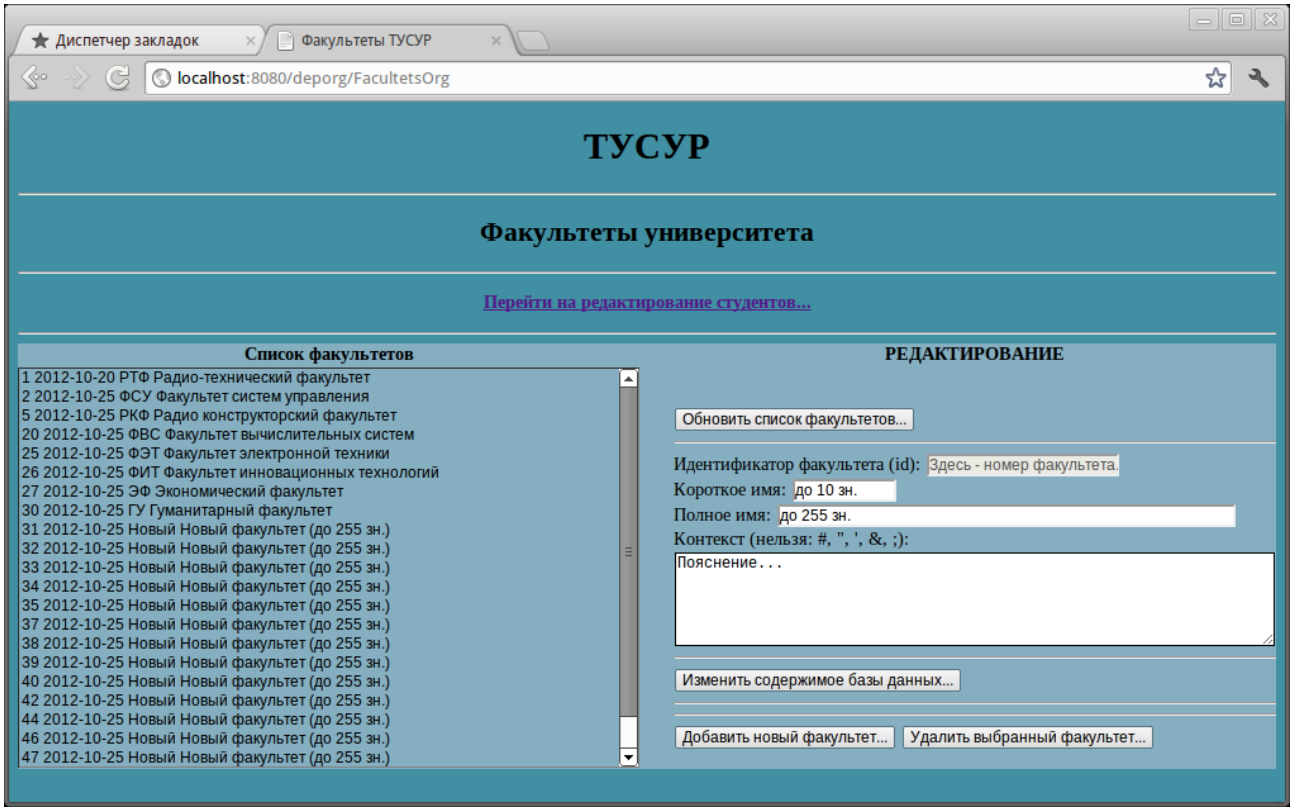

Рис. 6.2. Пример интерфейса приложения

Листинг 6.5. - Пример формы страницы

```
<FORM NAME=editFacultet METHOD="POST" ACTION="FacultetsOrg">
      <INPUT type=hidden name="idFacultet"><br>
      <INPUT type=submit VALUE="Обновить список факультетов..."><br>
      <hr>
      Идентификатор факультета (id):
      <INPUT name="idFac" value="Здесь - номер факультета..." disabled><br>
      Короткое имя: <input type="text" name="shortName" size="10"
            maxlength="10" value="до 10 зн."><br>
      Полное имя:  <input type="text" name="fullName" size="50"
            maxlength="255" value="до 255 зн."><br>
      Контекст (нельзя: #, ", ', &, ;):<br>
      <textarea rows="5" cols="65" name="Content">Пояснение...</textarea>
      <hr>
      <INPUT type=button name="updateFac" VALUE="Изменить содержимое базы
данных..."
            onClick="updateFacultet();">
      <hr><hr>
      <INPUT type=button name="insertFac" VALUE="Добавить новый факультет..."
            onClick="insertFacultet();">
      <INPUT type=button name="deleteFac" VALUE="Удалить выбранный факультет..."
            onClick="deleteFacultet();">
</FORM>
```

Основная проблема при передаче запроса от браузера к серверу состоит в модификации значений специальных символов и символов, имеющих кодовое

значение больше 127, в специальный формат, который по-разному может выполняться разными браузерами.

Указанная проблема эффективно устраняется применением следующих правил:

- в html-страницах следует использовать кодировку UTF-8;
- передаваемые браузером серверу значения параметров, содержащих русские буквы, кодировать с помощью функции `encodeURIComponent()`, входящей в язык JavaScript;
- на стороне сервлета, соответствующие параметры декодировать методом `URLDecoder.decode(arg, "UTF-8")`, входящим в пакет **java.net**.

6.3 Реализация функциональной части ССОД

Реализация функциональной части ССОД состоит в последовательном программировании сервлета и jsp-страницы, которое обеспечивает нужные действия с базой данных. Минимальный набор таких действий соответствует четырем действиям, осуществляемым с таблицами баз данных: **select**, **insert**, **delete** и **update**.

Лучшим объяснением таких действий, является демонстрация кода приложения.

Листинг 6.6 демонстрирует код jsp-страницы **facultets.jsp**.

Листинг 6.7 демонстрирует код сервлета **FacultetsOrg**.

Листинг 6.6. - Код интерфейса facultet.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Факультеты ТУСУР</title>

<style type="text/css">
<!--растягиваем список на всю ячейку-->
button{width:100%}
select{width:100%; background-color:#85afc1}
</style>

</head>
<%-- Объявления!!!
В этой части задается декларация всех переменных java.
--%>
<%!
//Значения параметров при вызове методами doGet() и doPost()
```

```
private String ss       = "";      //Рабочая переменная
private String [] slist = null;    //Рабочая переменная

//listFromDb - список записей факультетов
private String [] listFromDb = null;
//idFromDb - список идентификаторов факультетов
private String [] idFromDb = null;
//comment - коментарий к списоку факультетов
private String [] content = null;

%>
<%-- Установка начальных значений !!!
--%>
<%
String ss = (String)request.getAttribute("listFacultets");
if (ss != null) {
      listFromDb = ss.split(";"); //Список факультетов
      idFromDb   = ss.split(";");
      content    = ss.split(";");
      for (int i = 0; i < listFromDb.length; i++){
            slist = listFromDb[i].split("&");
            listFromDb[i] = slist[0];
            content[i]    = slist[1];
            slist = listFromDb[i].split("#");
            idFromDb[i] = slist[0];
      }
}
%>
<%-- Используем JavaScript для функциональной поддержки кнопок
--%>
<script type="text/javascript">
<%--Переменная для сохранения выбранного факультета--%>
var selectedFacultet = null;

<%--Проверка, что факультет выбран--%>
function isSelect(){
      if (selectedFacultet == null){
            alert('Выбери факультет!!!'); return false;
      }else return true;
}

<%--Сохраняю факультет при изменении в списке--%>
function saveFacultet(){
      selectedFacultet = document.listFacultets.facultet.value;
      getFacultet();
<%--  alert('Выбрано:\n' + selectedFacultet);
--%>
}

<%--Устанавливаю поля факультета для редактирования--%>
function getFacultet(){
      if (isSelect()){
            var aa = eval('lineFacultet' + selectedFacultet);
<%--        alert('Факультет - выбран!!!:\n' + aa[0] + '\n' +
                        aa[1] + '\n' +
                        aa[2] + '\n' +
                        aa[3] + '\n');
--%>
            document.editFacultet.idFacultet.value = aa[0];
            document.editFacultet.idFac.value      = aa[0];
            document.editFacultet.shortName.value  = aa[2];
```

```
            document.editFacultet.fullName.value   = aa[3];
            document.editFacultet.Content.value    = eval('lineContent' + selectedFacultet);
      }
}

<%--Запрос на добавление нового факультета--%>
function insertFacultet(){
      document.outPut.setaction.value = "insert";
      document.outPut.submit();
}

<%--Запрос на удаление факультета с заданным номером--%>
function deleteFacultet(){
      if (isSelect()){
            getFacultet();
            if( !confirm('Вы действительно хотите удалить факультет?:\nid = ' +
                        document.editFacultet.idFacultet.value)) return;
            var aa = eval('lineFacultet' + selectedFacultet);
            document.outPut.parameter.value = aa[0];
            document.outPut.setaction.value = "delete";
            document.outPut.submit();
      }
}

<%--Запрос на модификацию факультета с заданным номером--%>
function updateFacultet(){
      if (isSelect()){
         if( !confirm('Вы действительно хотите модифицировать факультет?:\nid = ' +
                        document.editFacultet.idFacultet.value)) return;

            var bb = document.editFacultet.idFacultet.value + '#' +
            encodeURIComponent('Это дата, она - по умолчанию') + '#' +
            encodeURIComponent(document.editFacultet.shortName.value) + '#' +
            encodeURIComponent(document.editFacultet.fullName.value) + '#' +
                  encodeURIComponent(document.editFacultet.Content.value);
            document.outPut.parameter.value = bb;
            document.outPut.setaction.value = "update";
            document.outPut.submit();
      }
}

<%--Сохраняю список факультетов в виде набора списков--%>
<%
for (int i = 0; i < listFromDb.length; i++){
      ss = listFromDb[i].replaceAll("#", "','");
      out.println("var lineFacultet" + idFromDb[i] + " = ['" + ss + "'];");
      out.println("var lineContent"  + idFromDb[i] + " = '" + content[i] +
"';");
}
%>
</script>

<body LANG="ru-RU" BGCOLOR="#418fa2">
<h1 align="center">ТУСУР</h1><hr>
<h2 align="center">Факультеты университета</h2>
<hr>
<%--
Меню-ссылка
--%>
<P ALIGN=CENTER >
<a HREF="StudentsOrg"><b>Перейти на редактирование студентов...</b></a>
</P>
```

```
<hr>
<TABLE WIDTH=100% CELLPADDING=0 CELLSPACING=0 BORDER=0 bgcolor="#85afc1">
<COLGROUP>
  <COL width="0*">
  <COL width="30">
  <COL width="500">
</COLGROUP>
<tr>
      <th>Список факультетов</th>
      <th></th>
      <th>РЕДАКТИРОВАНИЕ</th>
</tr>
<tr>
      <td>
            <FORM NAME=listFacultets METHOD="POST" ACTION="FacultetsOrg">
            <SELECT NAME="facultet" SIZE=21
                  onChange="saveFacultet();">
                  <%
                  if (listFromDb != null){
                        for (int i = 0; i < listFromDb.length; i++){
                              out.println("<option value=\"" + idFromDb[i] + "\">" +
                                    listFromDb[i].replaceAll("#", " ") + "</option>");
                        }
                  }
                  %>
            </SELECT><br>
            </FORM>
      </td>
      <td></td>
      <td>
            <FORM NAME=editFacultet METHOD="POST" ACTION="FacultetsOrg">
            <INPUT type=hidden name="idFacultet"><br>
            <INPUT type=submit VALUE="Обновить список факультетов..."><br>
            <hr>
            Идентификатор факультета (id):
            <INPUT name="idFac" value="Здесь - номер факультета..." disabled><br>
            Короткое имя: <input type="text" name="shortName" size="10"
                  maxlength="10" value="до 10 зн."><br>
            Полное имя:  <input type="text" name="fullName" size="50"
                  maxlength="255" value="до 255 зн."><br>
            Контекст (нельзя: #, ", ', &, ;):<br>
            <textarea rows="5" cols="65" name="Content">Пояснение...</textarea>
            <hr>
            <INPUT type=button name="updateFac" VALUE="Изменить содержимое базы данных..."
                  onClick="updateFacultet();">
            <hr><hr>
            <INPUT type=button name="insertFac" VALUE="Добавить новый факультет..."
                  onClick="insertFacultet();">
            <INPUT type=button name="deleteFac" VALUE="Удалить выбранный факультет..."
                  onClick="deleteFacultet();">
            </FORM>
      </td>
</tr>
</TABLE>
<%--
Скрытая форма для передачи запроса
--%>
      <FORM NAME=outPut METHOD="POST" ACTION="FacultetsOrg">
            <INPUT type=hidden name="setaction"      value="none" >
            <INPUT type=hidden name="parameter"><br>
      </FORM>
</body>
```

```
</html>
```

Листинг 6.7. - Код сервлета FacultetsOrg

```
package ru.tusur.department;

import java.io.*;
import java.net.URLDecoder;
//import java.net.URLEncoder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

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;

@WebServlet("/FacultetsOrg")

//ОСНОВНОЙ КЛАСС СЕРВЛЕТА
public class FacultetsOrg extends HttpServlet {
      private static final long serialVersionUID = 1L;

      //ОБЩИЕ ПАРАМЕТРЫ (.bashrc, /etc/host)
      /*
export URL_OK="jdbc:derby://dep_ok:1527/db_dep_ok;"
export URL_PREP="jdbc:derby://dep_prep:1527/db_dep_prep;"
export URL_DISC="jdbc:derby://dep_disc:1527/db_dep_disc;"
export URL_DECAN="jdbc:derby://dep_decan:1527/db_dep_decan;"
export ADD_ADM="user=userdep;password=userdep;"
       */
      private String URL_OK     = null;
      private String URL_PREP   = null;
      private String URL_DISC   = null;
      private String URL_DECAN  = null;
      private String ADD_ADM    = null;

      private String  URL_ADM      = null;
      private boolean isParameters = true;

    public FacultetsOrg() {
        super();
        //Читаем переменные среды
        URL_OK    = System.getenv("URL_OK");
        URL_PREP  = System.getenv("URL_PREP");
        URL_DISC  = System.getenv("URL_DISC");
```

```
    URL_DECAN = System.getenv("URL_DECAN");
    ADD_ADM   = System.getenv("ADD_ADM");

    if (URL_OK != null && ADD_ADM != null)
        URL_ADM = URL_OK + ADD_ADM;
}

//БАЗОВЫЕ МЕТОДЫ doGet() и doPost()
  protected void doGet(HttpServletRequest request,
        HttpServletResponse response)
              throws ServletException, IOException {

        doPost(request, response);
  }

  protected void doPost(HttpServletRequest request,
        HttpServletResponse response)
              throws ServletException, IOException {

        System.out.println("doPost(): URL_ADM = " + URL_ADM);
        //Проверяю параметры
        if (!testParameters(request)){
              //Вызов jsp-страницы с сообщением об ошибке
              RequestDispatcher disp =
                    request.getRequestDispatcher("/WEB-INF/errParameters.jsp");
              disp.forward(request, response);
              return;
        }

        //Читаю параметр setaction
        String act = request.getParameter("setaction");
        if (act != null)
        {
              if (act.equals("insert")){
                    insertFacultet(request);
              }
              if (act.equals("delete")){
                    deleteFacultet(request);
              }
              if (act.equals("update")){
                    updateFacultet(request);
              }
        }

        //Передаю список факультетов и
        //передаю его jsp-странице
        setListFacultets(request);

        //Вызов jsp-страницы
        RequestDispatcher disp =
                    request.getRequestDispatcher("/WEB-INF/facultets.jsp");
        disp.forward(request, response);
  }

  //Тест параметров среды
  protected boolean testParameters(HttpServletRequest request)
              throws ServletException, IOException{
        String er = "";
        if (URL_OK == null){
              isParameters = false;
              er = "URL_OK...\n";
        }
```

```
		if (URL_PREP == null){
			isParameters = false;
			if (er.length() > 0) er += "#";
			er += "URL_PREP...\n";
		}
		if (URL_DISC == null){
			isParameters = false;
			if (er.length() > 0) er += "#";
			er += "URL_DISC...\n";
		}
		if (URL_DECAN == null){
			isParameters = false;
			if (er.length() > 0) er += "#";
			er += "URL_DECAN...\n";
		}
		if (ADD_ADM == null){
			isParameters = false;
			if (er.length() > 0) er += "#";
			er += "ADD_ADM...\n";
		}
		if (!isParameters) request.setAttribute("errParameters", er);
		return isParameters;
	}

	//МЕТОДЫ ПОДДЕРЖКИ ФУНКЦИОНАЛЬНОСТИ СЕРВЛЕТА

	//Модифицируем выбранный факультет
	//
	protected void updateFacultet(HttpServletRequest request)
				throws ServletException, IOException{

		try{
			//Подключаем необходимый драйвер
			Class.forName("org.apache.derby.jdbc.ClientDriver");

			//Устанавливаем соединение с БД
			Connection conn = DriverManager.getConnection(URL_ADM);

			System.out.println("deleteFacultet: Connection - OK!");

			//Открываем объект запроса
			Statement st = conn.createStatement();


			//Делаем запрос
			String bb = request.getParameter("parameter");
			System.out.println("updateFacultet: parameter = " + bb);
			if (bb == null) return;
			String [] arg = bb.split("#");
			if (arg.length < 5) return;

		String req = "UPDATE facultets SET date_mod = DEFAULT, shortName = '";

			String dd = URLDecoder.decode(arg[2], "UTF-8");//Короткое имя
			if (dd.length() <= 0) dd = "DEFAULT";
			req += dd + "', fullName = '";

			dd = URLDecoder.decode(arg[3], "UTF-8");	//Длинное имя
			if (dd.length() <= 0) dd = "DEFAULT";
			req += dd + "', resume = '";

			dd = URLDecoder.decode(arg[4], "UTF-8");	//Контекст
```

```
            if (dd.length() <= 0) dd = "DEFAULT";
            req += dd + "' WHERE id = " + arg[0];

            System.out.println("updateFacultet: req = " + req);


            //Получаем ответ
            int nn =
                        st.executeUpdate(req);

      System.out.println("updateFacultet: БД open и update - OK!: " + nn + "\n");

            //Закрываем все объекты и разрываем соединение
            st.close();
            conn.commit();
            conn.close();

      }catch(ClassNotFoundException ex){System.out.println("updateFacultet:ex: " + ex);}
      catch(SQLException esql){System.out.println("updateFacultet:esql: " + esql);}
      catch(Exception e){System.out.println("updateFacultet:e: " + e);}

      finally{}
}

//Удаляем выбранный факультет
//
protected void deleteFacultet(HttpServletRequest request)
            throws ServletException, IOException{

      try{
            //Подключаем необходимый драйвер
            Class.forName("org.apache.derby.jdbc.ClientDriver");

            //Устанавливаем соединение с БД
            Connection conn = DriverManager.getConnection(URL_ADM);

            System.out.println("deleteFacultet: Connection - OK!");

            //Открываем объект запроса
            Statement st = conn.createStatement();


            //Делаем запрос
            String bb = request.getParameter("parameter");

            String req = "DELETE from facultets " +
                        "where id = " + bb;

            //Получаем ответ
            int nn =
                        st.executeUpdate(req);

      System.out.println("deleteFacultet: БД open и delete - OK!: " + nn + "\n");

            //Закрываем все объекты и разрываем соединение
            st.close();
            conn.commit();
            conn.close();

      }catch(ClassNotFoundException ex){System.out.println("deleteFacultet:ex: " + ex);}
      catch(SQLException esql){System.out.println("deleteFacultet:esql: " + esql);}
```

```
	catch(Exception e){System.out.println("deleteFacultet:e: " + e);}

	finally{}
}

//Добавляется новый факультет
//
protected void insertFacultet(HttpServletRequest request)
		throws ServletException, IOException{

	try{
		//Подключаем необходимый драйвер
		Class.forName("org.apache.derby.jdbc.ClientDriver");

		//Устанавливаем соединение с БД
		Connection conn = DriverManager.getConnection(URL_ADM);

		System.out.println("insertFacultet: Connection - OK!");

		//Открываем объект запроса
		Statement st = conn.createStatement();


		//Делаем запрос
	String req = "INSERT into facultets(date_mod, shortname, fullname, resume) " +
				"values(default,default,default,default)";

		//Получаем ответ
		int nn =
				st.executeUpdate(req);

	System.out.println("insertFacultet: БД open и insert - OK!: " + nn + "\n");

		//Закрываем все объекты и разрываем соединение
		st.close();
		conn.commit();
		conn.close();

	}catch(ClassNotFoundException ex){System.out.println("insertFacultet:ex: " + ex);}
	catch(SQLException esql){System.out.println("insertFacultet:esql: " + esql);}
	catch(Exception e){System.out.println("insertFacultet:e: " + e);}

	finally{}
}

//Чтение списка факультетов из таблицы app.v_facultets
//Создание атрибута listFacultets для передачи в jsp-страницу

protected void setListFacultets(HttpServletRequest request)
		throws ServletException, IOException {

	String ss = ""; //Исходный список записей
	try{
		//Подключаем необходимый драйвер
		Class.forName("org.apache.derby.jdbc.ClientDriver");

		//Устанавливаем соединение с БД
		Connection conn = DriverManager.getConnection(URL_OK);

		System.out.println("setListFacultets: Connection - OK! ");

		//Открываем объект запроса
```

```
                Statement st = conn.createStatement();


                //Делаем запрос
            String req = "SELECT CHAR(id), CHAR(date_mod), shortName, fullname, resume " +
                             "FROM v_facultets ORDER BY id";

                //Получаем ответ
                ResultSet rs =
                             st.executeQuery(req);

                //Формируем в строку результат запроса к БД
                String ss2 = "";
                while(rs.next()){
                      ss2 =   rs.getString(1).trim() + "#" +
                                  rs.getString(2) + "#" +
                                  rs.getString(3) + "#" +
                                  rs.getString(4) + "&" +
                                  rs.getString(5);
                      if (ss.length() > 0) ss += ";" + ss2;
                      else ss = ss2;
                }
            System.out.println("setListFacultets: БД create и select - OK!: " + ss + "\n");

                //Закрываем все объекты и разрываем соединение
                rs.close();
                st.close();
                conn.commit();
                conn.close();

            }catch(ClassNotFoundException ex){System.out.println("setListWrites:ex: " + ex);}
            catch(SQLException esql){System.out.println("setListWrites:esql: " + esql);}
            catch(Exception e){System.out.println("setListWrites:e: " + e);}
            finally{
                  if (ss.length() > 0) request.setAttribute("listFacultets",  ss);
            }
      }
}
```

Далее, обучающийся проводит разработку остальных функциональных частей локального приложения.

Преподаватель, по мере завершения отдельных частей практического занятия, оценивает объем и сложность выполненного задания.

ЛИТЕРАТУРА

1. Проект Apache Derby. - 7 с. (Файл: Derby1.pdf)
2. Введение в системы реляционных баз данных Derby. - 8 с. (Файл: Derby2.pdf)
3. Введение в JDBC Derby. - 8 с. (Файл: Derby3.pdf)
4. Архив официальной документации по СУБД Apache Derby (источники информации на английском языке). - (Файл архива: db-derby.tar)
5. Проект Eclipse. - 20 с. (Файл: Eclipse1.pdf)
6. Введение в интегрированную среду разработки Eclipse. - 12 с. (Файл: Eclipse2.pdf)
7. Разработка подключаемых модулей для Eclipse. - 12 с. (Файл: Eclipse3.pdf)
8. Еще раз о разработке плагинов Eclipse. - 21 с. (Файл: Eclipse4.pdf)

Учебное издание

Резник Виталий Григорьевич

СОВРЕМЕННЫЕ КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ

Методические рекомендации для практических занятий по дисциплине «Современные компьютерные технологии» для студентов уровня основной образовательной программы магистратура
направления подготовки 010400.68 «Прикладная математика и информатика»
профиля «Математическое и программное обеспечение вычислительных комплексов и компьютерных сетей».

Учебно-методическое пособие

Усл. печ. л. . Тираж ___. Заказ .

Томский государственный университет

систем управления и радиоэлектроники

634050, г. Томск, пр. Ленина, 40