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 курсовому проектированию

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 курсовому проектированию: Учебно-методическое пособие. – Томск, ТУСУР, 2012. – 48 с.

Учебно-методическое пособие предназначено для выполнения курсового проектирования 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 учебно-методическое пособие содержит методические рекомендации, обеспечивающие успешное курсовое проектирование по дисциплине «Современные компьютерные технологии» (СКТ).

Все работы по курсовому проектированию 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 прошел обучение по первой части дисциплины «Современные компьютерные технологии»;
- владеет теоретическими знаниями и практическим умением, полученными при изучении лекционного материала и 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 вопросу проектирования;
- завершает проектные работы письменным отчетом и реализацией программного обеспечения на ЭВМ.

Данное методическое пособие содержит девять разделов, охватывающих все вопросы курсового проектирования.

Первый раздел содержит описание общей тематики проектирования и ПО УПК АСУ, необходимое для выполнения проектных работ. Приведен список вариантов заданий на проектирование.

Разделы со второго по пятый содержат рекомендации и примеры проектирования теоретической части работы.

Разделы с шестого по восьмой содержат рекомендации и требования по выполнению практической части проекта.

Раздел девятый определяет правила и процедуру приема проекта в целом.

В конце методического пособия приведен список литературы для изучения теоретического материала и проведения проектных работ по данной дисциплине. Электронные варианты документов этого списка предоставляются в файлах формата pdf. Имеются также ссылки на сетевые информационные ресурсы.

1. Выбор темы курсового проекта. Определение задания на обзорную часть проекта

Общая тематика курсового проектирования предполагает создание отдельной части офисного распределенного приложения, содержащего:

- интерфейсную часть приложения, представляющую набор html-страниц, отображаемых браузером общего пользования;
- серверную часть приложения, представляющую набор сервлетов Apache Tomcat, установленного на ЭВМ студента;
- локальную базу данных СУБД Apache Derby, доступную с компьютеров локальной сети УПК АСУ и содержащую часть таблиц и представлений общего распределенного приложения.

Студент проводит проектирование приложения, используя:

- ПО УПК АСУ в составе, приведенном в подразделе 1.1;
- вариант темы и ограничения реализации приложения, приведенные в подразделе 1.2.

1.1 Состав ПО УПК АСУ для проведения проектных работ

Общий состав и функциональные возможности УПК АСУ изучены студентами при выполнении лабораторных работ по дисциплине «Современные операционные системы» и первой части дисциплины «Современные компьютерные технологии».

Необходимый набор дистрибутивов ПО для выполнения проектных работ представлен в таблице 1.

Таблица 1. Перечень дистрибутивов ПО УПК АСУ (/cdrom/casper/asu11upk/opt/)

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

В таблице 2 представлен необходимый для запуска перечень скриптов.

Таблица 2. Скрипты ПО УПК АСУ ($HOME/bin/)

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 рабочей среды пользователя (проектировщика), необходимо использовать файлы:

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

Для типовой реализации проектной работы, в домашней директории пользователя (проектировщика) 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.

1.2 Перечень тем курсового проектирования

Прикладная тематика курсового проектирования предполагает создание распределенной системы департаментов некоторого абстрактного вуза — «ТУСУР».

Каждый департамент имеет собственный локальный сервер, содержащий:

- базу данных СУБД Apache Derby, хранящую информацию о деятельности вуза 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.

Целью курсового проектирования является создание программного обеспечения отдельного департамента из списка представленного в таблице 3.

Функциональные возможности ПО департамента должны обеспечивать:

- возможность редактирования информации, содержащейся в базе данных департамента, уполномоченным сотрудником (проектировщиком);

- доступ к информации департамента (без возможности редактирования) всеми участниками проектов распределенной системы «ТУСУР».

Таблица 3. Темы курсового проектирования

№	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)

Минимальный состав информационной части проекта, отраженный в базах данных, приведен в таблице 4.

Таблица 4. 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.

Список имен баз данных, по темам основной части проекта, приведен в таблице 5.

Таблица 5. 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

2. Контроль и уточнение обзорной части проекта

Цель данной части проекта — формализация и согласование информационного содержания каждой части распределенного приложения.

Для этой цели проводятся:

- согласование и настройка сетевой среды каждой операционной системы локального приложения;
- определение и формализация таблиц баз данных каждого локального приложения.

2.1 Сетевая среда распределенного приложения

Сервер каждого локального приложения имеет:

- **IP-адрес**, по которому он доступен в сети;
- **номер порта** СУБД Apache Derby (**1527**) для 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 распределенного приложения, следует согласовать и настроить именование этих приложений.

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

Пример такой настройки приведен в листинге 1.

Листинг 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
#---------------------------------------
```

2.2 Формализация информационной части проекта

В пределах выбранной темы следует разработать информационную структуру таблиц, приведенных в таблице 4 для каждой части проекта.

Структура таблиц должна полностью удовлетворять требованиям всего распределенного приложения.

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

После согласования информационных структур баз данных, следует дать их формализованное описание на языке SQL.

Пример такого описания для таблиц **facultets** и **students**, а также их представлений **v_facultets** и **v_students**, для общедоступной схемы **APP**, приведен в листинге 2.

Листинг 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;
```

3. Обсуждение отчета по обзорной части проекта

В этой части проектных работ студенты должны:

- иметь полное теоретическое представление о выполняемой работе;
- создать в инструментальной среде Eclipse EE проект с именем **deport**;
- в рамках проекта **deport** создать необходимые базы данных и таблицы;
- иметь черновой вариант обзорной части проекта.

В пределах занятия студенты выполняют следующие работы:

- излагают и обсуждают доклады о проделанной части работы;
- вносят последние изменения и дополнения в структуру баз данных;
- обмениваются информацией о структурах баз данных;
- проводят взаимное тестирование доступа к публичным частям всех баз данных распределенного приложения.

Пример скрипта для тестирования базы данных **db_dep_ok** приведен в листинге 3.

Листинг 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;
------------------------------------------------------------
```

4. Оформление отчета по обзорной части проекта

На этапе оформления обзорной части проекта студент подводит итог теоретических изысканий в плане решения личной локальной задачи, а также фиксирует дополнения в плане решения всей задачи, в пределах всего распределенного приложения. Непосредственно должны быть выполнены:

- пояснительная записка проекта, содержащая письменное изложение решений, выполненных на предыдущих этапах проектирования;
- подготовлен план практической реализации локального проекта;
- решены и обоснованы способы взаимодействия с другими участниками общего проекта.

Пояснительная записка проекта должна содержать:

- Титульный лист проекта, с необходимыми атрибутами, определенными требованиями кафедры АСУ для работ данного типа;
- Лист задания проектирования, однозначно отображающий тему проекта и ограничительные требования на его исполнение;
- Лист оглавления пояснительной записки: «СОДЕРЖАНИЕ» с перечисленными разделами и подразделами документа, а также со ссылками страниц на их размещение;
- В содержательной части пояснительной записки должны быть выделены: «Введение», «Основная (содержательная) часть», «Заключение» и «Литература (список используемых источников)».

Формальное завершение теоретической (обзорной) части проектной работы завершается контрольным этапом, предполагающий прием обзорной части проекта преподавателем.

5. Прием отчета по обзорной части проекта

На этом этапе студент демонстрирует преподавателю:

- Пояснительную записку по обзорной части проекта;
- Понимание функционального назначения локальной части проекта;
- Понимание и способы взаимодействия с другими частями общего распределенного приложения;
- Готовность к выполнению практической части работы.

Замечание. Важным вопросом многих приложений рассматриваемого типа является наличие источников информации об адресах ресурсов, используемых локальным приложением. В рамках учебного проекта таким ресурсом может рассматриваться информационная среда самой операционной системы (переменные среды).

С целью однозначного толкования размещения общих информационных ресурсов проекта, студенту следует добавить в файл **.bashrc** переменные среды, на основе которых можно формировать URL-адреса и сопутствующую информацию поддержку при старте локальных приложений проекта.

Пример варианта задания переменных среды приведен в листинге 4.

Листинг 4. Общие параметры переменных среды

```
#-------------------------------------------------------------------------------
#ОБЩИЕ ПАРАМЕТРЫ (.bashrc)
#-------------------------------------------------------------------------------
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;"
#-------------------------------------------------------------------------------
```

Замечание. Следует обратить особое внимание на наличие «кавычек» при задании строковых переменных, содержащих символ «точка с запятой».

После редактирования файла **.bashrc,** следует выполнить команду: **putdbin**, чтобы сохранить и использовать, в дальнейшем, все проведенные изменения.

6. Выбор контрольного примера по проекту

На этапе выбора контрольного примера по проекту, студентом решаются следующие задачи:

- определение состава интерфейсной и функциональной частей проекта, выраженной в перечислении jsp-страниц, необходимых для реализации интерфейсов пользователя, и состава сервлетов, обеспечивающих функциональную поддержку интерфейсов;
- определение средств обработки ошибок, возникающих при отсутствии или неправильном задании параметров среды приложения.

6.1 Определение состава интерфейсной и функциональной частей проекта

Определение состава интерфейсной и функциональной частей проекта приведем на примере локальной задачи: **dep_ok** - «Департамент студенческого отдела кадров» (см. таблицы 3-5).

Согласно поставленной задаче, данный проект должен обеспечить модификацию списка факультетов и списка студентов, распределяемых по факультетам.

Для решения поставленной задачи достаточно двух сервлетов и двух jsp-страниц, приведенных в таблице 6.

Таблица 6. 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 таблице 6 программные объекты создаются в среде разработки Eclipse EE проекта **deport**. На листинге 5 показан первоначальный шаблон сервлета **StudentsOrg**, вызывающий jsp-страницу **students.jsp**.

Листинг 5. 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 приведен шаблон jsp-страницы для работы со списком студентов, а на рис. 1 — отображение этой страницы в браузере.

Листинг 6. 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>
```

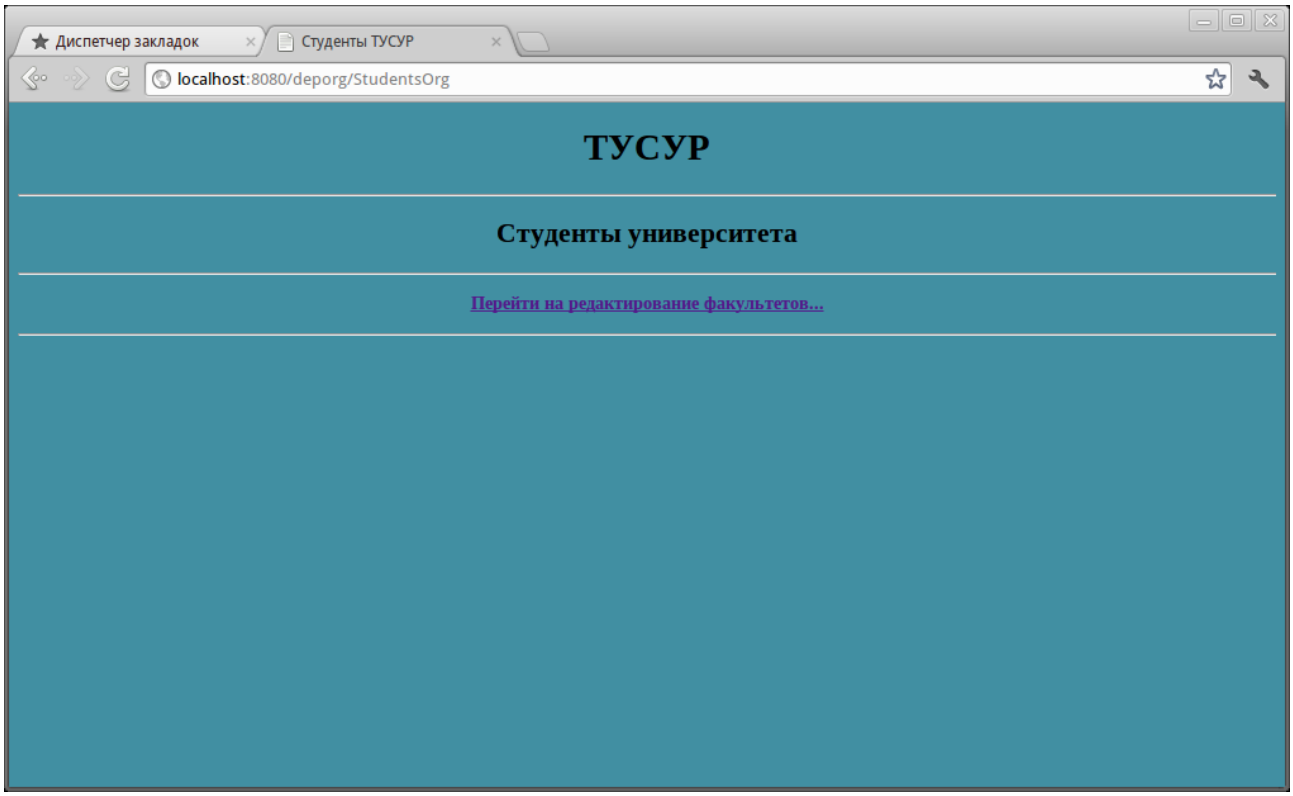

Рис. 1. Отображение шаблона jsp-страницы students.jsp

6.2 Обработка ошибок чтения параметров среды приложения

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 этих параметров 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.

Листинг 7 показывает решение этого вопроса для сервлета **FacultetsOrg**.

Листинг 8 показывает jsp-страницу, отображающую обнаруженные ошибки.

Листинг 7. 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 8. 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>
```

7. Реализация контрольного примера

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.

7.1 Подготовка интерфейсной части проекта

Подготовка интерфейсной части проекта состоит в программировании jsp-страниц на языке html с целью размещения на странице, отображаемой браузером, 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. 2.

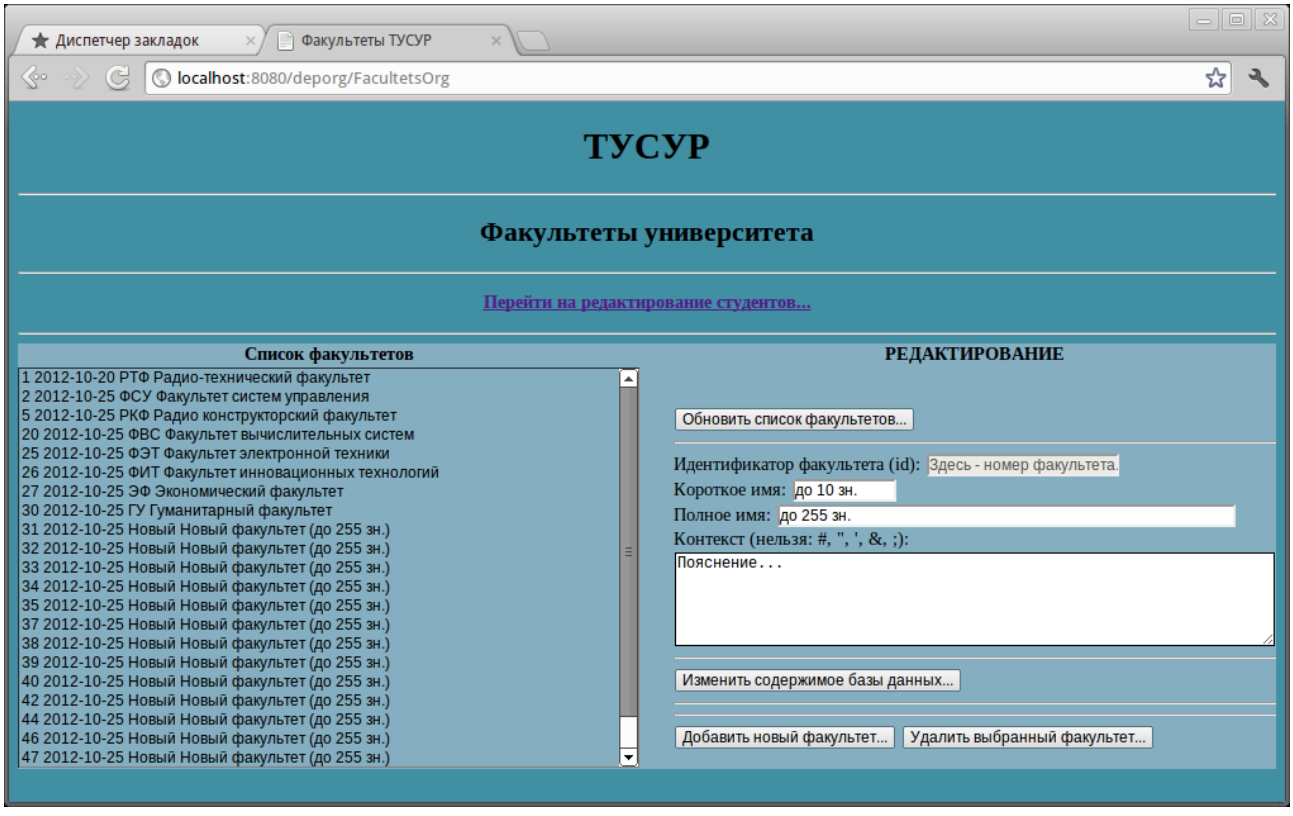

Рис. 2. Пример интерфейса приложения

7.2 Передача параметров запроса формами из html-страниц

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».

Пример такой формы, соответствующей части интерфейса, показанного на рис. 2, приведен в листинге 9.

Листинг 9. 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**.

7.3 Последовательная реализация функциональной части проекта

Реализация функциональной части проекта 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 10 демонстрирует код jsp-страницы **facultets.jsp**.

Листинг 11 демонстрирует код сервлета **FacultetsOrg**.

Листинг 10. 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 11. 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");

            //Устанвливаем 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");

            //Устанвливаем 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);
            }
      }
}
```

7.4 Оптимизация проекта: кэширование запросов к БД

После реализации функциональной части проекта, следует убедиться в его полном соответствии ограничительным требованиям задания. Это соответствие является минимально необходимым условием для сдачи проекта.

Дополнительно, следует провести целостный анализ проекта для выявления качеств реализации проекта, непосредственно не предусмотренные его ограничительной частью. Такими качествами, для рассматриваемого типа проектов, являются:

1. Необходимость и достаточность обращений интерфейсной части приложения к информации и функциям серверной части приложения;
2. Ограничительные возможности приложения в плане адекватного функционирования и допустимые возможности расширения этих границ.

Хотя подобный анализ допускает значительную степень субъективизма, в ряде случаев удается значительно улучшить качество проекта или, в крайнем случае, определить проблемные области использования проекта. В качестве примера такого анализа рассмотрим реализацию интерфейсной и функциональной части задачи редактирования списка студентов. На рис. 3 представлен один из возможных вариантов интерфейсной части задачи.

Видно, что основная компоновка интерфейса в целом соответствует аналогичному интерфейсу для редактирования списка факультетов:

- в левой части имеется список объектов редактирования;
- в правой части — форма для редактирования полей выбранного объекта.

Отличительной особенностью отображения списка студентов является контекстная зависимость от значения факультета, представленная двумя выпадающими меню с «длинным» и «коротким» названиями факультета.

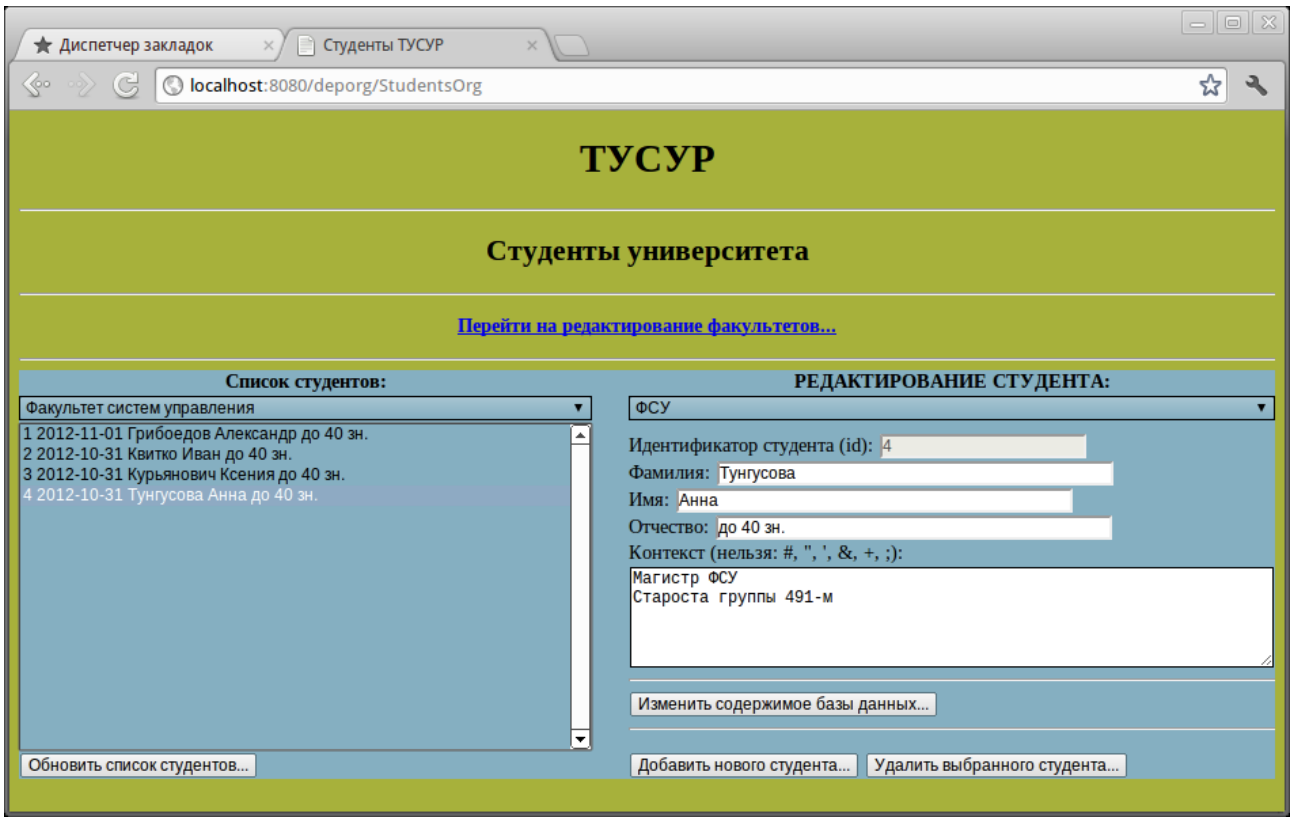

Рис. 3. Интерфейс для редактирования списка студентов

Алгоритмическую формализацию данной особенности можно записать следующим образом:

- если изменяется название факультета, то следует сделать запрос к серверу на получение нового списка студентов;
- в пределах монопольного доступа к департаменту отдела кадров студента и в пределах интерфейса редактирования списка студентов, список факультетов есть постоянная величина.

В пределах рассматриваемой части проекта, программируемыми объектами являются:

- **StudentsOrg.java** — сервлет, обеспечивающий функциональную часть приложения;
- **students.jsp** — интерфейсная страница списка студентов.

Поскольку в пределах использования страницы **students.jsp**, список факультетов не изменяется, то сервлету **StudentsOrg.java** достаточно запросить этот список при старте, сформировать коды выпадающих меню для «длинных» и «коротких» имен факультетов, хранить эти коды до соответствующих запросов браузера и передавать их в jsp-страницу — по-необходимости.

Реализация такого сервлета представлена в листинге 12. Обратите внимание, что списки факультетов хранятся в строковых переменных **shortList** и **fullList**, а для их создания используется отдельный метод **listFacultets(...)**.

Листинг 12. Реализация сервлета StudentsOrg

```
package ru.tusur.department;

import java.io.IOException;
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({ "/StudentsOrg", "/" })

//ОСНОВНОЙ КЛАСС СЕРВЛЕТА
public class StudentsOrg 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;

      //Списки факультетов в готовой форме
      String shortList = "";
      String fullList  = "";

      public StudentsOrg() {
        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")){
                  insertStudent(request);
            }
            if (act.equals("delete")){
                  deleteStudent(request);
            }
            if (act.equals("update")){
                  updateStudent(request);
            }
            if (act.equals("reset")){
                  listFacultets(request);
            }
      }

      //Формирую списоки факультетов и студентов
      //Передаю их jsp-странице
      if (shortList.length() <= 0 || fullList.length() <= 0) listFacultets(request);

      listStudents(request);

      //Вызов jsp-страницы
      RequestDispatcher disp =
                  request.getRequestDispatcher("/WEB-INF/students.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 данные выбранного студента
//
protected void updateStudent(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("updateStudent: Connection - OK!");

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


              //Делаем запрос
              String bb = request.getParameter("parameter");
              System.out.println("updateStudent: parameter = " + bb);
              if (bb == null) return;
              String [] arg = bb.split("#");
              if (arg.length < 6) return;

              String req = "UPDATE students SET idf = " + arg[1];
                          req += ", date_mod = DEFAULT, family = '";

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

              dd = URLDecoder.decode(arg[4], "UTF-8");       //Имя
              if (dd.length() <= 0) dd = "DEFAULT";
              req += dd + "', fathername = '";

              dd = URLDecoder.decode(arg[5], "UTF-8");       //Отчество
              if (dd.length() <= 0) dd = "DEFAULT";
              req += dd + "', resume = '";

              dd = URLDecoder.decode(arg[6], "UTF-8");       //Контекст
```

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

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


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

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

//Удаляем выбранного студента
//
protected void deleteStudent(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("deleteStudent: Connection - OK!");

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


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

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

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

            System.out.println("deleteStudent: 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("deleteStudent:ex: " + ex);}
      catch(SQLException esql){System.out.println("deleteStudent:esql: " + esql);}
      catch(Exception e){System.out.println("deleteStudent:e: " + e);}
      finally{}
```

```
}

//Добавляется новый Студент
//
protected void insertStudent(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("insertStudent: Connection - OK!");

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


            //Делаем запрос
String req = "INSERT into students(idf, date_mod, family, name, fathername, resume) " +
                        "values(default,default,default,default,default,default)";

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

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

//Чтение списка факультетов из таблицы app.v_facultets

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

      shortList = "<option value=\"0\" selected>Новый</option>\n";
      fullList  = "<option value=\"0\">Новый студент</option>\n";

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

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

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

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

```
            //Делаем запрос на список факультетов
            String req = "SELECT CHAR(id), shortName, fullname " +
                        "FROM v_facultets ORDER BY shortName";

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

            //Формируем в строку результат запроса к БД
            String ss2 = "";
            while(rs.next()){
                  ss2 =   rs.getString(1).trim();
shortList += "<option value=\"" + ss2 + "\" >" + rs.getString(2) + "</option>\n";
fullList  += "<option value=\"" + ss2 + "\" >" + rs.getString(3) + "</option>\n";
            }

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

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

//Создание атрибута listFacultets для передачи в jsp-страницу

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

      String selectedFacultet = request.getParameter("selectedFacultet");
      if (selectedFacultet == null) selectedFacultet = "0";
      request.setAttribute("selectedFacultet", selectedFacultet);
      request.setAttribute("shortListFacultets", shortList);
      request.setAttribute("fullListFacultets",  fullList);

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

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

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

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


            //Делаем запрос на список студентов
      String req = "SELECT CHAR(id), CHAR(date_mod), family, name, fathername, resume " +
            "FROM  v_students WHERE idf = " + selectedFacultet + " ORDER BY family";

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

            //Формируем в строку результат запроса к БД
            String ss = ""; //Исходный список записей
            String ss2 = "";
            while(rs.next()){
```

```
                        ss2 =   rs.getString(1).trim() + "#" +
                                    rs.getString(2) + "#" +
                                    rs.getString(3) + "#" +
                                    rs.getString(4) + "#" +
                                    rs.getString(5) + "&" +
                                    URLEncoder.encode(rs.getString(6),"UTF-8");
                        if (ss.length() > 0) ss += ";";
                        ss += ss2;
                    }
            System.out.println("listStudents: БД create и select students - OK!: " + ss + "\n");
                    if (ss.length() > 0) request.setAttribute("listStudents",  ss);

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

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

Замечание. Кэширование всегда вызывает скрытые проблемы, о которых проектировщик приложений всегда должен помнить. В нашем примере явно присутствуют две проблемы:

1. Кэширование страницы браузером, что устраняется принудительной перезагрузкой пользователем вызванной страницы.
2. Переход со страницы редактирования факультетов на страницу редактирования студентов требует обновления списка факультетов, чтобы правильно отображать внесенные изменения. Решение этой проблемы показано в листинге 13. Здесь в ссылке кода **facultets.jsp** передается параметр для сервлета **StudentsOrg**, требующий обновления списка факультетов. Соответственно, сервлет **StudentsOrg** должен распознать этот параметр и выполнить действия, например, как в коде листинга 12.

Листинг 13. Код страницы facultets.jsp, передающей параметр по ссылке

```
<hr>
<%--
Меню-ссылка
--%>
<P ALIGN=CENTER >
<a HREF="StudentsOrg?setaction=reset">
<b>Перейти на редактирование студентов...</b>
</a>
</P>
<hr>
```

7.5 Оптимизация проекта: кодирование текстовой информации

Проектировщику постоянно приходится решать проблемы ограничений при использовании национальных языков в современных технологиях:

- браузеры преобразуют параметры запроса к серверу, использующие специальные символы и буквы национальных кодировок;
- языки программирования как на стороне сервера, так и на стороне браузера, также чувствительны к специальным кодировкам.

Если в пределах одного языка программирования указанные проблемы решаются более или менее приемлемо, то в распределенной среде с разными языками программирования, для получения удовлетворительного результата требуются специальные приемы.

Для примера, рассмотрим интерфейс редактирования списка студентов, показанный на рис. 3. Поле в нижней правой части рисунка, названное «**Контекст**», должно содержать произвольный текст, состоящий из множества строк. Когда в левой части интерфейса выбирается конкретный студент, поля правой части интерфейса заполняются средствами языка JavaScript на основе сохраненной информации.

Хотя языки java и JavaScript используют одну кодировку Unicode, текст, содержащий множество строк и представленый строкой языка JavaScript, не может быть обработан некоторыми методами этого языка, например, **eval(...)**.

Одним из способов решения этой проблемы является кодирование сервлетом этого текста перед отправкой браузеру и последующим декодированием текста средствами языка JavaScript перед записью нужное поле формы. В листинге 14 представлен участок кода сервлета **StudentsOrg**, который выполняет эту операцию.

Листинг 14. Кодирование текста в методе listStudents(...)

```
//Формируем в строку результат запроса к БД
String ss = ""; //Исходный список записей
String ss2 = "";
while(rs.next()){
      ss2 =   rs.getString(1).trim() + "#" +
                  rs.getString(2) + "#" +
                  rs.getString(3) + "#" +
                  rs.getString(4) + "#" +
                  rs.getString(5) + "&" +
                  URLEncoder.encode(rs.getString(6),"UTF-8");
      if (ss.length() > 0) ss += ";";
      ss += ss2;
}
```

В листинге 15 представлен полный текст кода страницы **students.jsp**, обеспечивающий требуемое решение задачи.

Листинг 15. Полный код 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>
<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

//shortListFacultets - список (коротких имен) факультетов
private String shortListFacultets = null;
//fullListFacultets - список (длинных имен) факультетов
private String fullListFacultets = null;
//selectedFacultet - выбранное длинное имя факультета
private String selFac = "0";

//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("shortListFacultets");
if (ss != null) shortListFacultets = ss;
else shortListFacultets = "<option value=\"0\" selected>Новый</option>";

ss = (String)request.getAttribute("fullListFacultets");
if (ss != null) fullListFacultets = ss;
else fullListFacultets = "<option value=\"0\" selected>Новые студенты</option>";

ss = (String)request.getAttribute("selectedFacultet");
if (ss != null) selFac = ss;

ss = (String)request.getAttribute("listStudents");
```

```
listFromDb = null;
if (ss != null) {
      if (ss.length() > 0){
            listFromDb = ss.split(";"); //Список студентов
            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 выбранных:
факультета и студента--%>
var selectedFacultet = <%=selFac%>;
var selectedStudent  = null;

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

<%--После загрузки страницы--%>
function afterStart(){
<%--        alert('После загрузки страницы!!!\nselectedFacultet = ' +
                  selectedFacultet + ' -- ' + <%=selFac%>);
--%>
            document.fullListFacultets.fullFacultet.value = selectedFacultet;
}

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

<%--Сохраняю студента при изменении в списке--%>
function saveStudent(){
      selectedStudent = document.listStudents.student.value;
      getStudent();
<%--  alert('Выбрано:\n' + selectedStudent);
--%>
}

<%--Устанавливаю поля студента для редактирования--%>
function getStudent(){
      if (isSelect()){
```

```
            var aa = eval('lineStudent' + selectedStudent);
            document.editStudent.idStud.value     = aa[0];
            document.editStudent.family.value     = aa[2];
            document.editStudent.name.value       = aa[3];
            document.editStudent.fathername.value = aa[4];

            var bb = decodeURIComponent(eval('lineContent' + selectedStudent));
            aa = bb.split('+');
            document.editStudent.Content.value    = aa.join(' ');
            selectedFacultet = document.fullListFacultets.fullFacultet.value;
            document.shortListFacultets.shortFacultet.value = selectedFacultet;

      }
}

<%--Запрос на получение нового списка--%>
function getNewList(){
      selectedFacultet = document.fullListFacultets.fullFacultet.value;
      document.outPut.selectedFacultet.value = selectedFacultet;
      document.outPut.setaction.value = "none";
      document.outPut.submit();
}

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

<%--Запрос на удаление факультета с заданным номером--%>
function deleteStudent(){
      if (isSelect()){
            getStudent();
            if( !confirm('Вы действительно хотите удалить студента?:\nid = ' +
                        document.editStudent.idStud.value)) return;
            var aa = eval('lineStudent' + selectedStudent);
            document.outPut.parameter.value = aa[0];
            document.outPut.setaction.value = "delete";
            document.outPut.submit();
      }
}

<%--Запрос на модификацию факультета с заданным номером--%>
function updateStudent(){
      if (isSelect()){
            if( !confirm('Вы действительно хотите модифицировать данные
студента?:\nid = ' +
                        document.editStudent.idStud.value)) return;
            var bb = document.editStudent.idStud.value + '#' +
            document.shortListFacultets.shortFacultet.value + '#' +
            encodeURIComponent('Это дата, она - по умолчанию') + '#' +
            encodeURIComponent(document.editStudent.family.value) + '#' +
            encodeURIComponent(document.editStudent.name.value) + '#' +
            encodeURIComponent(document.editStudent.fathername.value) + '#' +
                  encodeURIComponent(document.editStudent.Content.value);
            document.outPut.parameter.value = bb;
            document.outPut.setaction.value = "update";
            document.outPut.selectedFacultet.value =
                  document.shortListFacultets.shortFacultet.value;
            document.outPut.submit();
      }
}
```

```
</script>

<body LANG="ru-RU" BGCOLOR="#a7b13b" onLoad="afterStart();">
<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>
<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 студентов:
            <FORM NAME="fullListFacultets" METHOD="POST" ACTION="StudentsOrg">
            <SELECT NAME="fullFacultet" onChange="getNewList();">
                  <%=fullListFacultets%>
            </SELECT><br>
            </FORM>
      </th>
      <th></th>
      <th>РЕДАКТИРОВАНИЕ СТУДЕНТА:
            <FORM NAME="shortListFacultets" METHOD="POST" ACTION="StudentsOrg">
            <SELECT NAME="shortFacultet" onChange="saveFacultet();">
                  <%=shortListFacultets%>
            </SELECT><br>
            </FORM>
      </th>
</tr>
<tr>
      <td>
            <FORM NAME="listStudents" METHOD="POST" ACTION="StudentsOrg">
            <SELECT NAME="student" SIZE=16
                  onChange="saveStudent();">
                  <%
                  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="editStudent" METHOD="POST" ACTION="StudentsOrg">
            Идентификатор студента (id):
            <INPUT name="idStud" value="Здесь - номер студента..." disabled><br>
            Фамилия: <input type="text" name="family" size="40" maxlength="40"
                  value="до 40 зн."><br>
            Имя: <input type="text" name="name"  size="40" maxlength="40" value="до 40 зн."><br>
            Отчество: <input type="text" name="fathername" size="40" maxlength="40"
                  value="до 40 зн."><br>
                  Контекст (нельзя: #, ", ', &, +, ;):<br>
            <textarea rows="5" cols="65" name="Content">Пояснение... до 4096 зн.</textarea>
            <hr>
```

```
            <INPUT type=button name="updateStud" VALUE="Изменить содержимое базы данных..."
                        onClick="updateStudent();">
            <hr>
            </FORM>
      </td>
</tr>
<tr>
      <td>
                  <FORM NAME="newList" METHOD="POST" ACTION="StudentsOrg">
                  <INPUT type=button VALUE="Обновить список студентов..."
                        onClick="getNewList();">
                  </FORM>
      </td>
      <td></td>
      <td>
            <FORM NAME="insertAndDelete" METHOD="POST" ACTION="StudentsOrg">
            <INPUT type=button name="insertStud" VALUE="Добавить нового студента..."
                  onClick="insertStudent();">
            <INPUT type=button name="deleteStud" VALUE="Удалить выбранного студента..."
                  onClick="deleteStudent();">
            </FORM>
      </td>
</tr>
</TABLE>
      <FORM NAME="outPut" METHOD="POST" ACTION="StudentsOrg">
                  <INPUT type=hidden name="setaction"      value="none">
                  <INPUT type=hidden name="parameter"      value="none">
                  <INPUT type=hidden name="selectedFacultet" value="0">
      </FORM>
</body>
</html>
```

Замечание. Обратите внимание, что функция **decodeURIComponent(...)** языка JavaScript не декодирует символ «+», который представляет пробел в кодируемом тексте. Этот факт следует учесть в реализации проекта.

В листинге 16 приведена функция, заполняющая необходимые поля студента, предварительно проведя декодирование поля **Content**.

Листинг 16. Код функции getStudent()

```
<%--Устанавливаю поля студента для редактирования--%>
function getStudent(){
      if (isSelect()){
            var aa = eval('lineStudent' + selectedStudent);
            document.editStudent.idStud.value     = aa[0];
            document.editStudent.family.value     = aa[2];
            document.editStudent.name.value       = aa[3];
            document.editStudent.fathername.value = aa[4];

            var bb = decodeURIComponent(eval('lineContent' + selectedStudent));
            aa = bb.split('+');
            document.editStudent.Content.value    = aa.join(' ');
            selectedFacultet = document.fullListFacultets.fullFacultet.value;
            document.shortListFacultets.shortFacultet.value = selectedFacultet;

      }
}
```

8. Обсуждение практической части курсового проекта

Целями данного этапа проектирования является:

- формирование у студентов навыков коллективной разработки проектов;
- своевременное выявление и устранение проблем, влияющих на выполнение общей части проекта;
- текущий контроль выполнения заданий студентами.

По мере завершения разработки отдельных функциональных частей локального приложения, студент должен дополнять основную часть пояснительной записки к проекту.

Преподаватель по мере завершения отдельных частей проекта оценивает объем и сложность выполненного задания.

9. Прием курсового проекта

Прием курсового проекта производится преподавателем на основе персонального доклада студентом итогов выполненных работ. Обязательными составляющими доклада являются:

- работоспособное локальное приложение, спроектированное студентом в соответствии с его персональным заданием;
- наличие полностью оформленной пояснительной записки, представленной в отпечатанном, переплетенном и подписанном студентом виде.
- Наличие на документе пояснительной записки проекта оценки и подписи преподавателя.

По завершению приема курсового проекта, пояснительная записка по проекту передается преподавателю и остается у него.

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 по курсовому проектированию 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