Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение наукии высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники**

**(ИИТР)**

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 03**

**по дисциплине**

**«Распределенные программные системы»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Трифонов Д.А.

Владимир, 2020

ЦЕЛЬ РАБОТЫ

Освоить создание классов сущностей, в том числе путем обратного проектирования реляционной базы данных средствами NetBeans 5.5 и работу с менеджером сущностей. и получить навыки работы со средой разработки.

ЗАДАНИЕ

1. Создать не менее двух связанных классов сущностей с помощью обратного проектирования базы данных.

2. Разработать веб-приложение для редактирования сущностей, созданных с помощью обратного проектирования.

3. Разработать не менее двух связанных между собой классов сущностей вручную.

4. Использовать следующие типы связей: один ко многим, многие ко многим.

5. Использовать автоматическую генерацию первичных ключей.

6. Разработать класс-обработчик событий ЖЦ сущности, который позволяет проследить действия менеджера сущностей со всеми сущностями, разработанными в пп.1 и 2.

7. Изменить сессионный компонент таким образом, чтобы вместо прямого доступа к базе данных через интерфейс JDBC использовался менеджер сущностей, а разработанные сущности использовались в качестве параметров и результатов бизнес-методов сессионных компонентов.

8. Определить не менее одного запроса, использующего не менее двух сущностей и использовать его в коде сессионного компонента.

9. Доработать тестовое клиентское приложение, вызывающее бизнес-методы сессионных компонентов.

10. Показать успешную установку Java EE-приложения, создание таблиц для сущностей и выполнение тестового клиента.

ВЫПОЛНЕНИЕ РАБОТЫ

Первым шагом выполнения лабораторной работы будет создание веб-приложение для создания и редактирования сущностей встроенными средствами NetBeans 5.5. Создаем простое веб приложение и настраиваем подключение к существующей бд. (Рисунок 1)

С помощью мастера создания сущностей из таблиц базы данных получаем класс сущности user. (Рисунок 2) Далее автоматически создаем сеансовый компонент и страницы JSF для классов сущностей. Результат создания приведен на рисунке 3. Запуск и проверка работоспособности приложения – рисунки 4-6.

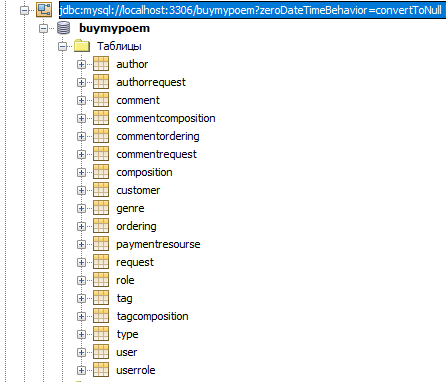


Рисунок 1.Добавили новое подключение. (Бд по курсовому проекту)

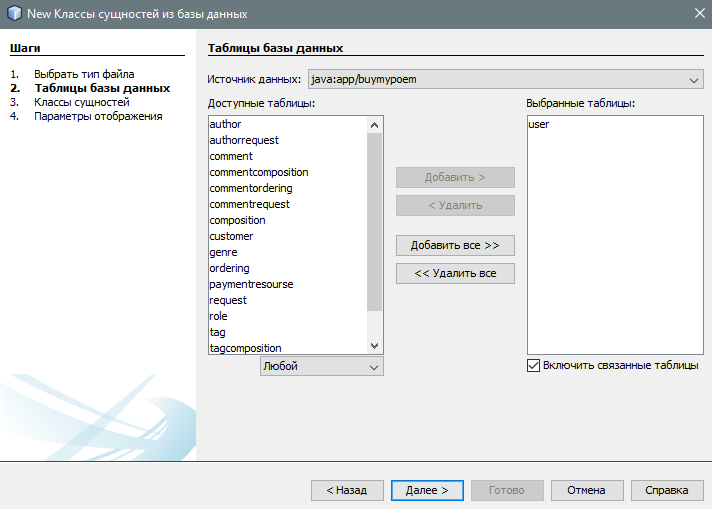


Рисунок 2. Создание класса сущности из базы данных.

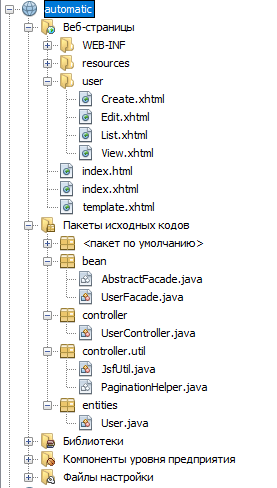


Рисунок 3. Автоматически созданные компоненты веб приложения.

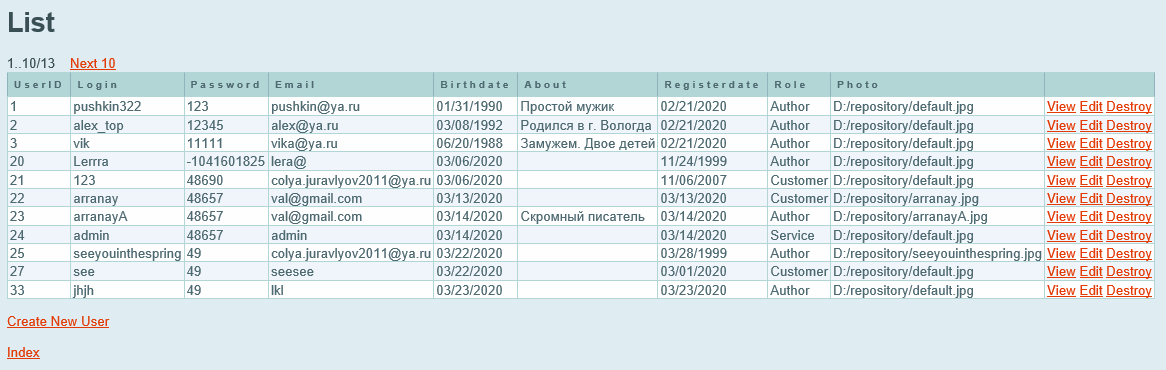


Рисунок 4. Вывод всех пользователей.

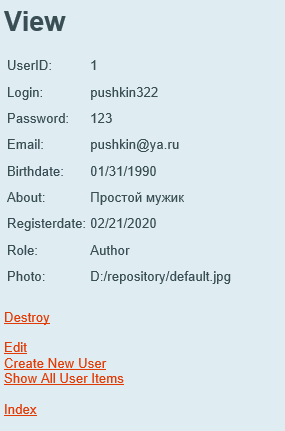


Рисунок 5. Просмотр информации о конкретном пользователе.

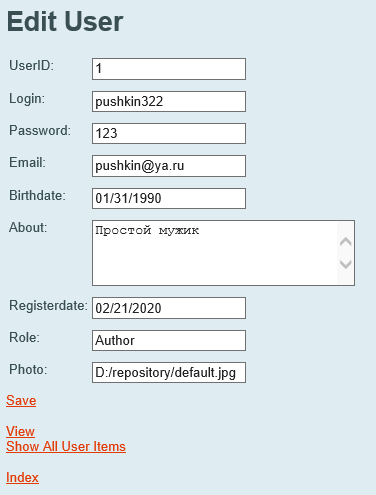


Рисунок 6. Изменение информации о пользователе.

|  |
| --- |
|  |

Далее, создаем новое подключение и новый проект. Создаем классы сущностей User и Role, добавляем единицу персистентности. В созданные классы добавляем новые свойства, гет и сет методы и конструктор. Классы имеют связь один-ко-многим.

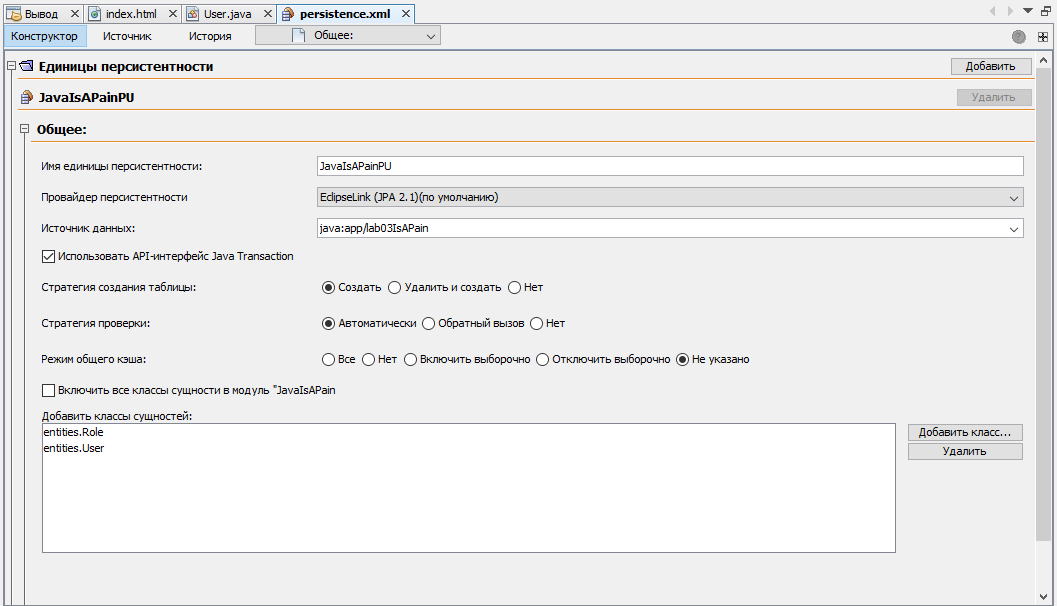


Рисунок 7. Редактор файла persistence.xml

<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_1.xsd">

<persistence-unit name="JavaIsAPainPU" transaction-type="JTA">

<jta-data-source>java:app/lab03IsAPain</jta-data-source>

<class>entities.Role</class>

<class>entities.User</class>

<exclude-unlisted-classes>true</exclude-unlisted-classes>

<properties>

<property name="javax.persistence.schema-generation.database.action" value="create"/>

</properties>

</persistence-unit>

</persistence>

Класс сущности - User:

package entities;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

@Entity

@Table(name = "user")

public class User implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Integer id;

@Column(name = "login")

private String login;

@Column(name = "password")

private String password;

@JoinColumn(name = "role", referencedColumnName = "id")

@ManyToOne(optional = false)

private Role role;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getLogin() {

return login;

}

public void setLogin(String login) {

this.login = login;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Role getRole() {

return role;

}

public void setRole(Role role) {

this.role = role;

}

public User() {

}

public User(Integer id) {

this.id = id;

}

public User(Integer id, String login, String password) {

this.id = id;

this.login = login;

this.password = password;

}

}

Класс сущности - Role:

package entities;

import java.io.Serializable;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "role")

public class Role implements Serializable {

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy = GenerationType.AUTO)

private Integer id;

@Column(name = "name")

private String name;

@Column(name = "description")

private String description;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDescription() {

return description;

}

public void setDescription(String description) {

this.description = description;

}

public Role(){

}

public Role(Integer id, String name, String description) {

this.id = id;

this.name = name;

this.description = description;

}

public Role(Integer id) {

this.id = id;

}

}

Создаем сессионные компоненты без состояния с локальным интерфейсом. Добавляем ссылку на менеджер сущностей. Через менеджер сущностей реализовываем CRUD операции.

Локальный интерфейс компонента User:

package bean;

import entities.User;

import java.util.List;

import javax.ejb.Local;

@Local

public interface UserSessionBeanLocal {

public List<User> findAll();

public List<User> findAllStudent();

public void create(User u);

public void edit(User u);

public void remove(User u);

public User find(int id);

}

Сессионный компонент User без сохранения состояния:

package bean;

import entities.User;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class UserSessionBean implements UserSessionBeanLocal {

@PersistenceContext(unitName = "JavaIsAPainPU")

private EntityManager em;

@Override

public List<User> findAll() {

List <User> userList

=em.createQuery("SELECT u FROM User u").getResultList();

return userList;

}

@Override

public List<User> findAllStudent() {

List <User> userList

=em.createQuery("select u from User u WHERE u.role.name='student'").getResultList();

return userList;

}

@Override

public void create(User u) {

em.persist(u);

}

@Override

public void edit(User u) {

em.merge(u);

}

@Override

public void remove(User u) {

em.remove(em.merge(u));

}

@Override

public User find(int id) {

return em.find(User.class, id);

}

}

Локальный интерфейс компонента Role:

package bean;

import entities.Role;

import java.util.List;

import javax.ejb.Local;

@Local

public interface RoleSessionBeanLocal {

public List<Role> findAll();

public Role find(int id);

}

Сессионный компонент Role без сохранения состояния:

package bean;

import entities.Role;

import java.util.List;

import javax.ejb.Stateless;

import javax.persistence.EntityManager;

import javax.persistence.PersistenceContext;

@Stateless

public class RoleSessionBean implements RoleSessionBeanLocal {

@PersistenceContext(unitName = "JavaIsAPainPU")

private EntityManager em;

@Override

public List<Role> findAll() {

List <Role> roleList

=em.createQuery("SELECT r FROM Role r").getResultList();

return roleList;

}

@Override

public Role find(int id) {

Role role = em.find(Role.class, id);

return role;

}

}

Контроллер для вывода всех пользователей и удаления конкретного пользователя:

package controllers;

import bean.UserSessionBeanLocal;

import entities.User;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class userList extends HttpServlet {

@EJB

UserSessionBeanLocal userSBL;

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

try{

List<User> userList = userSBL.findAll();

request.setAttribute("userList", userList);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("userList.jsp");

requestDispatcher.forward(request, response);

}catch (Exception e) {

throw new ServletException(e.getMessage());

}

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int id = Integer.parseInt(request.getParameter("id"));

User u = userSBL.find(id);

userSBL.remove(u);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("index.jsp");

requestDispatcher.forward(request, response);

}

}

Контроллеры для вывода только студентов и вывода всех ролей выглядят аналогично, без метода doPost.

JSP вывода всех пользователей:

<%@ page contentType="text/html;charset=UTF-8" language="java" %>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Список пользователей</title>

<style>

table{

border-spacing:0;

width: 80%;

height: 100%;

font-size: 20px;

}

.btn{

background: none;

color: black;

border: 0;

font-size: 20px;

}

</style>

</head>

<body>

<table align="center">

<tr>

<td>логин</td>

<td>пароль</td>

<td>роль в системе</td>

<td></td>

<td></td>

</tr>

<c:forEach items="${userList}" var="user">

<tr>

<td>${user.login}</td>

<td>${user.password}</td>

<td>${user.role.name}</td>

<td>

<form action="editUser" method="get">

<input type="hidden" name="id" value="${user.id}">

<input class="btn" type="submit" value="редактировать">

</form>

</td>

<td><form action="userList" method="post">

<input type="hidden" name="id" value="${user.id}">

<input class="btn" type="submit" value="удалить">

</form>

</td>

</tr>

</c:forEach>

</table>

</body>

</html>

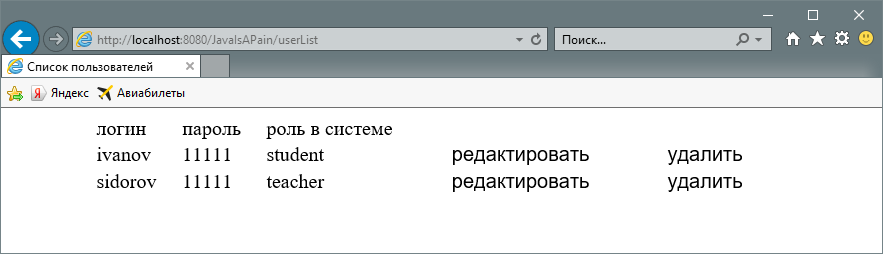


Рисунок 8. Вывод всех пользователей.

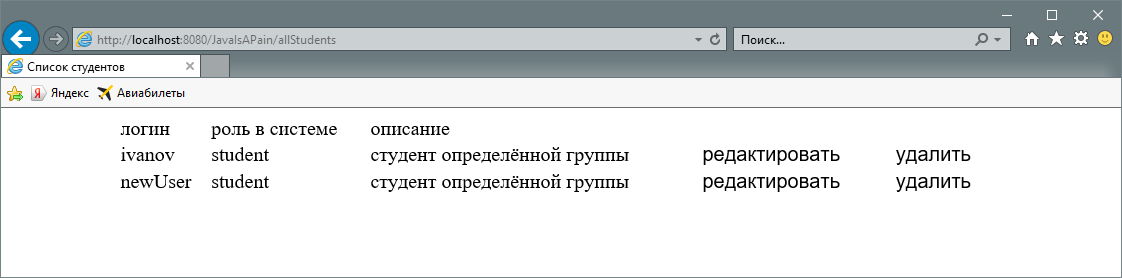


Рисунок 9. Пример выполнения именованного запроса с выбором всех студентов.

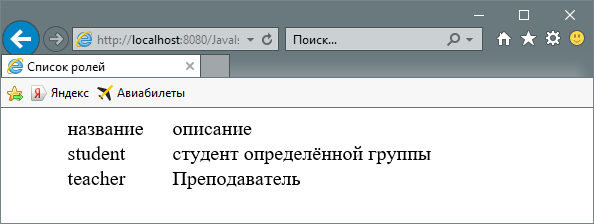


Рисунок 10. Вывод списка ролей.

Контроллер для добавления нового пользователя:

package controllers;

import bean.RoleSessionBeanLocal;

import bean.UserSessionBeanLocal;

import entities.Role;

import entities.User;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class addUser extends HttpServlet {

@EJB

RoleSessionBeanLocal roleSBL;

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

try{

List<Role> roleList = roleSBL.findAll();

request.setAttribute("roleList", roleList);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("addUser.jsp");

requestDispatcher.forward(request, response);

}catch (Exception e) {

throw new ServletException(e.getMessage());

}

}

@EJB

UserSessionBeanLocal userSBL;

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

if (!(request.getParameter("password").equals(request.getParameter("password2")))){

request.setAttribute("error", "Пароли не совпадают");

List<Role> roleList = roleSBL.findAll();

request.setAttribute("roleList", roleList);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("addUser.jsp");

requestDispatcher.forward(request, response);

}

User user = new User();

user.setLogin(request.getParameter("login"));

user.setPassword(request.getParameter("password"));

Role role = roleSBL.find(Integer.parseInt(request.getParameter("role")));

user.setRole(role);

userSBL.create(user);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("index.jsp");

requestDispatcher.forward(request, response);

}

}

JSP страница для добавления нового пользователя:

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Создание нового пользователя</title>

<style>

.error{

color: red;

}

</style>

</head>

<body>

<form action="addUser" method="post">

<table>

<tr>

<td>Логин</td>

<td><input required type="text" name="login" /></td>

</tr>

<tr>

<td>Пароль</td>

<td><input required type="password" name="password" /></td>

</tr>

<tr>

<td>Повторите пароль</td>

<td><input required type="password" name="password2" /></td>

</tr>

<tr>

<td>Роль</td>

<td><select name="role">

<c:forEach items="${roleList}" var="role">

<option value="${role.id}">${role.name}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td>

<input required type="submit" value="Сохранить"/>

</td></tr>

</table>

</form>

<h3 class="error">${error}</h3>

</body>

</html>

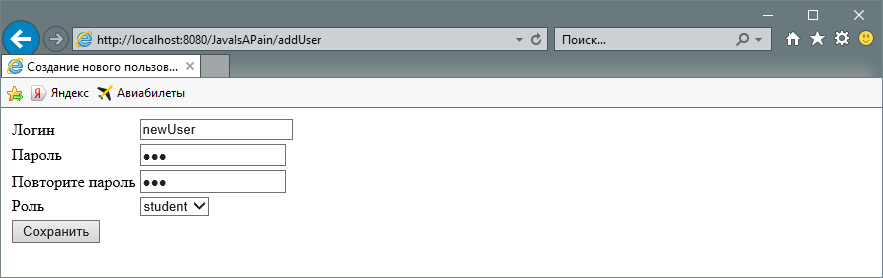


Рисунок 11. Добавление нового пользователя.

Контроллер для редактирования данных о пользователе:

package controllers;

import bean.RoleSessionBeanLocal;

import bean.UserSessionBeanLocal;

import entities.Role;

import entities.User;

import java.io.IOException;

import java.util.List;

import javax.ejb.EJB;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

public class editUser extends HttpServlet {

@EJB

UserSessionBeanLocal userSBL;

@EJB

RoleSessionBeanLocal roleSBL;

User u;

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

int id = Integer.parseInt(request.getParameter("id"));

u = userSBL.find(id);

request.setAttribute("user", u);

List<Role> roleList = roleSBL.findAll();

request.setAttribute("roleList", roleList);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("editUser.jsp");

requestDispatcher.forward(request, response);

}

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

u.setLogin(request.getParameter("login"));

Role role = roleSBL.find(Integer.parseInt(request.getParameter("role")));

u.setRole(role);

userSBL.edit(u);

RequestDispatcher requestDispatcher = request.getRequestDispatcher("index.jsp");

requestDispatcher.forward(request, response);

}

}

JSP страница для редактирования:

<%@page contentType="text/html" pageEncoding="UTF-8"%>

<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>Редактировать</title>

<style>

</style>

</head>

<body>

<form action="editUser" method="post">

<input type="hidden" name="id" value="${user.id}">

<table>

<tr> <td>текущий логин - ${user.login}</td>

<td><input required type="text" name="login" /></td>

</tr>

<tr><td>текущая роль - ${user.role.name}</td>

<td><select name="role">

<c:forEach items="${roleList}" var="role">

<option value="${role.id}">${role.name}</option>

</c:forEach>

</select></td>

</tr>

<tr>

<td colspan="2"><input type="submit" value="сохранить"></td>

</tr></table>

</form>

</body>

</html>

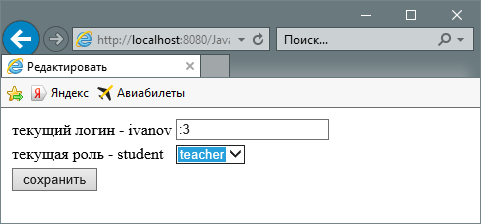


Рисунок 12.Страница редактирования.

ВЫВОД

В ходе выполнения лабораторной работы было освоено создание классов сущностей, в том числе путем обратного проектирования реляционной базы данных средствами NetBeans 5.5, работа с менеджером сущностей и получены навыки работы со средой разработки. Deploy приложения был выполнен на сервер Glassfish 4.1.1.