Studentų 50 – 213
Kompiuterių katedra

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Informatikos inžinerijos studijų programa

Laboratorinis darbas Nr. 1

T120B143 Įmonių kompiuterinių sistemų kūrimo platformos

Įmonės kompiuterinės sistemos realizacija panaudojant Servlet, JBeans, Java Persistence ir JSP technologijas

ATLIKO:

Ignas Jasonas _____ **IFF-6/6**
(Vardas Pavardė) (Parašas) (Grupė)

DĖSTYTOJAS:

doc. A. Liutkevičius _____
(Vardas Pavardė) (Parašas)

DARBAS ATIDUOTAS:

05 d. 11 mėn. 2019

KAUNAS, 2019

Įmonės kompiuterinės sistemos realizacija panaudojant Servlet, JBeans, Java Persistence ir JSP technologijas

Užduotis

- Sukurti vartotojo sąsają panaudojant JavaServer Pages;
- Sukurti logikos sluoksnį panaudojant Servlet;
- Sukurti JavaBean kuriame gali būti laikinai saugomi duomenys;
- Sukurti EJB (Enterprise JavaBean) panaudojant Java Persistence, kuris saugomas duomenų bazėje.

1 laboratorinio darbo sprendimo architektūra/aprašymas

Uždaviniams realizuoti buvo naudojamas vitualioje Windows XP mašinoje pateiktas pranešimų saugojimo sistemos pavyzdys.

Vartojo sąsaja sukurta panaudojant JavaServer Pages, kurioje vartotojas turi galimybę matyti įrašytus pranešimus, taip pat sukurti savo pranešimą su pasirinktu vardu bei žinute. Taip pat yra įgyvendintas paskutinio įrašyto pranešimo rodymas.

Logikos sluoksniui realizuoti panaudotas Servlet. Šis sluoksnis apdoroja kreipinius, ateinančius iš vartotojo naršyklės. Šiuo atveju realizuoti dviejų kreipinių apdorojimai – GET kreipinys, kuris grąžina visus pranešimus ir POST kreipinys, kuris išsaugo pranešimą į duombazę.

Logikos sluoksnyje, pradžioje inicijuojamas JavaBean pranešimo klasės objektas, kuriame bus laikinai saugomas paskutinis vartotojo įrašytas pranešimas.

Panaudojant Java Persistance technologiją realizuota pranešimo esybės klasė, aprašanti sąryšį tarp pranešimo klasės sąryšį su duomenų bazės lentele.

Šiai sistemai paleisti naudojamas GlassFish 4.0 serveris

Naudojama Apache Derby duomenų bazė

1 laboratorinio darbo sprendimo programinis kodas

JavaBeans klasė

Message.java

```
/*

 * To change this template, choose Tools | Templates

 * and open the template in the editor.

 */

package jlab1.beans;



import java.util.Date;
```

```
/**
 *
 * @author Administrator
 */
public class Message {

    public Message() {
    }

    private String name;
    private String msg;
    private Date time;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

    public Date getTime() {
```

```
        return time;
    }

    public void setTime(Date time) {
        this.time = time;
    }
}
```

Esybės klasė

Message.java

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package jlab1.entities;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;
import javax.persistence.TableGenerator;
import javax.persistence.Temporal;
import javax.persistence.TemporalType;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import javax.xml.bind.annotation.XmlRootElement;

/**
 *
 * @author Administrator
 */
@Entity
@Table(name = "MESSAGE")
@XmlRootElement
@NamedQueries({
    @NamedQuery(name = "Message.findAll", query = "SELECT m FROM Message m"),
    @NamedQuery(name = "Message.findById", query = "SELECT m FROM Message m WHERE m.id =
:id"),
    @NamedQuery(name = "Message.findByName", query = "SELECT m FROM Message m WHERE
m.name = :name"),
    @NamedQuery(name = "Message.findByMessage", query = "SELECT m FROM Message m WHERE
m.message = :message"),
    @NamedQuery(name = "Message.findByTime", query = "SELECT m FROM Message m WHERE m.time
= :time")})
public class Message implements Serializable {
    private static final long serialVersionUID = 1L;
```

```
@Id
@Basic(optional = false)
@NotNull
@TableGenerator(name = "MSGPkGen",
        table = "SEQUENCE",
        schema = "APP",
        pkColumnName = "SEQ_NAME",
        pkColumnValue = "MESSAGE",
        valueColumnName = "SEQ_COUNT",
        initialValue = 0,
        allocationSize = 1)
@GeneratedValue(generator = "MSGPkGen", strategy = GenerationType.TABLE)
@Column(name = "ID")
private Integer id;
@Size(max = 50)
@Column(name = "NAME")
private String name;
@Size(max = 120)
@Column(name = "MESSAGE")
private String message;
@Column(name = "TIME")
@Temporal(TemporalType.TIME)
private Date time;

public Message() {
}

public Message(Integer id) {
    this.id = id;
}

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 getMessage() {
    return message;
}

public void setMessage(String message) {
    this.message = message;
}

public Date getTime() {
    return time;
}

public void setTime(Date time) {
```

```
        this.time = time;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (id != null ? id.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id fields are not set
        if (!(object instanceof Message)) {
            return false;
        }
        Message other = (Message) object;
        if ((this.id == null && other.id != null) || (this.id != null && !this.id.equals(other.id))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "jlab1.entities.Message[ id=" + id + " ]";
    }

}
```

Servlet klasė

Namai.java

```
/*

 * To change this template, choose Tools | Templates

 * and open the template in the editor.

 */

package jlab1.servlets;



import java.io.IOException;

import java.util.Date;

import java.util.List;

import javax.annotation.Resource;

import javax.persistence.EntityManager;

import javax.persistence.EntityManagerFactory;

import javax.persistence.PersistenceUnit;

import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.transaction.UserTransaction;
import jlab1.beans.Message;

/**
 *
 * @author Administrator
 */
public class Namai extends HttpServlet {

    private jlab1.beans.Message msg = new Message();
    @PersistenceUnit
    private EntityManagerFactory emf;
    @Resource
    private UserTransaction utx;

    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
```

```
    EntityManager em = null;


    try {
        em = emf.createEntityManager();


        List messages = em.createQuery("SELECT m FROM Message m").getResultList();
        request.setAttribute("msg", this.msg);
        request.setAttribute("msg_list", messages);


        request.getRequestDispatcher("namai.jsp").forward(request, response);
    } catch (Exception ex) {
        throw new ServletException(ex);
    } finally {
        if (em != null) {
            em.close();
        }
    }
}


    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to
edit the code.">
    /**
     * Handles the HTTP
     * <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

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

/**
 * Handles the HTTP
 * <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

    String l_name = "";
    l_name = request.getParameter("name");

    String l_msg = "";
    l_msg = request.getParameter("message");

    if (l_name != null && l_msg != null) {
        this.msg.setName(l_name);
        this.msg.setMsg(l_msg);
        this.msg.setTime(new Date());

        EntityManager em = null;
        try {
            jlab1.entities.Message e_msg = new jlab1.entities.Message();
```

```
            e_msg.setName(l_name);
            e_msg.setMessage(l_msg);
            e_msg.setTime(new Date());

            utx.begin();
            em = emf.createEntityManager();
            em.persist(e_msg);
            utx.commit();

        } catch (Exception ex) {
            throw new ServletException(ex);
        } finally {
            if (em != null) {
                em.close();
            }
        }

    }


    processRequest(request, response);
}

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo() {
    return "Short description";
```

```
    }// </editor-fold>
}
```

JavaServer Pages failas

Namai.jsp

```
%--
    Document   : namai
    Created on : Oct 29, 2014, 11:16:08 AM
    Author     : Administrator
--%>

<%@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=ISO-8859-1">
        <title> Messages List </title>
        <style>
            TABLE, TD, TH, TR {
                border-collapse:collapse;
                border-width: 1px;
                border-style: solid;
                border-spacing: 5px;
                padding: 2px 5px;
            }
        </style>

    </head>
    <body align="center">
        <form action="/jlab1" method="POST">
            Vardas
```

```
    <input type="text" name="name" size="20px"/>
    Komentaras
    <input type="text" name="message" size="20px"/>
    <input type="submit" value="Siųsti"/>
</form>
<hr>
<div>
    <c:if test="${not empty msg}">
        <jsp:getProperty name="msg" property="name"/>:
        <jsp:getProperty name="msg" property="msg"/>
    </c:if>
</div>
<hr>
<div align="center">
    <table id="messageTable">
        <tr>
            <th>ID</th>
            <th>Vardas</th>
            <th>Komentaras</th>
            <th>Data</th>
        </tr>
        <c:if test="${not empty msg_list}">
            <c:forEach var="m" items="${msg_list}">
                <tr>
                    <td>${m.id}  </td>
                    <td>${m.name}  </td>
                    <td>${m.message}  </td>
                    <td>${m.time}  </td>
                </tr>

            </c:forEach>
```

```
            </c:if>
          </table>
        </div>
    </body>
</html>
```

Rezultatų apibendrinimas

Šio laboratorinio darbo metu buvo susipažinta su Java programavimo įrankiais, kurie skirti palengvinti įmonių sistemų kūrimo darbą, tai yra JavaServer Pages, Servlet ir Java Persistance. Šie įrankiai suteikia lengvesnį sistemos išplečiamumą, gerą dinamiškumą kuriant vartotojo sąsają bei patogesnį duomenų bazės valdymą.