Studentų 50 – 213

Kompiuterių katedra

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:**

**Tadas Laurinaitis**  **IFF – 6/8**

(Vardas Pavardė) (Parašas) (Grupė)

**DĖSTYTOJAS:**

**doc. A. Liutkevičius**

(Vardas Pavardė) (Parašas)

**DARBAS ATIDUOTAS:**

\_\_\_d.\_\_\_\_\_\_\_mėn. 2019

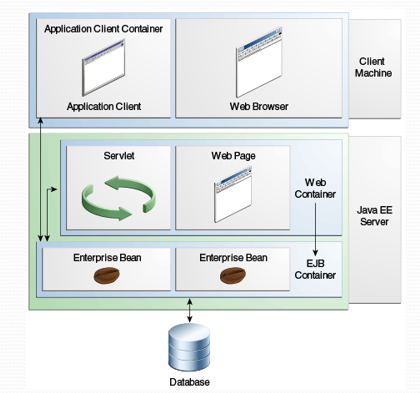
**KAUNAS, 2019**

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

# Užduotys

* 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.

# XXX sprendimo architektūra/aprašymas



An application client container (ACC) - is a set of Java classes, libraries and other files required for application client execution which are bundled with the application client for distribution.

web browser (commonly referred to as a browser) is a software application for accessing information on the World Wide Web. When a user opens a particular website, the web browser retrieves the necessary content from a web server then displays the resulting web page on the user's device.

A web container (also known as a servlet container; and compare "webcontainer") is the component of a web server that interacts with Java servlets.

Java servlet is a Java software component that extends the capabilities of a server. Although servlets can respond to many types of requests, they most commonly implement web containers for hosting web applications on web servers and thus qualify as a server-side servlet web API.

An enterprise bean is a Java component that can be combined with other resources to create Java applications. There are three types of enterprise beans, entity beans, session beans, and message-driven beans.

EJB is a server-side software component that encapsulates business logic of an application. An EJB web container provides a runtime environment for web related software components, including computer security, Java servlet lifecycle management, transaction processing, and other web services.

# Sprendimo programinis kodas

**Namai servlet klasė**

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;

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>

}

**Message bean klasė**

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;

@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 + " ]";

}

}

**Message entity klasė**

package jlab1.beans;

import java.util.Date;

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;

}

}

# Rezultatų apibendrinimas

Šiame skyriuje trumpai reziumuojama, kas buvo atlikta darbo metu, kokie projektiniai ir architektūriniai sprendimai pasirinkti, kitokiomis priemonėmis buvo atliktas projektavimas ir realizacija, įvertinamas šių priemonių efektyvumas ir t.t.

Darbo metu buvo padaryta:

* Sukurta minimali vartotojo sąsaja panaudojant JavaServer Pages
* Sukurtas logikos sluoksnis panaudojant Servlet
* Sukurti JavaBean kuriame gali būti laikainai saugomi duomenys
* Sukurti EJB panaudojant Java Persistence, kuris saugomas duomenų bazėje

Projektinių ir architektūrinių sprendimų neteko priimti, nes darbas buvo jau padarytas ir jį kažkaip kitaip perdarinėti nėra prasmės.