JAVA SERVER FACES
PRIMJERI DIJELOVA
SERVERSKIH APLIKACIJA

master EW-008.STUD-ETF-IP.046_AeroflotUsers / WebContent-Administration / WEB-INF / FRONTAL / **workers.jsp**

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

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

<c:out value='${workerAdminBean.initializeSession(pageContext.session)}'></c:out>

<c:if test='${param["w"] ne null}'>
        <c:out value='${workerAdminBean.set(param["w"])}'></c:out>
</c:if>

<span id='workers_form_target'></span>
<br>
АДМИНИСТРИРАЊЕ ПОДАЦИМА О ЗАПОСЛЕНИМ
<br><br>
```

```
<f:view>
        <h:form id="workers-form">
                <table>
                        <tr>
                                <td><h:outputLabel for="username-wk"> Корисничко име : </h:outputLabel></td>
                        </tr>
                <tr>
                        <td><h:inputText id="username-wk" value="#{workerAdminBean.userName}"/></td>
                        </tr>
                        <tr>
                                <td><h:outputLabel for="password-wk"> Лозинка : </h:outputLabel></td>
                </tr>
                <tr>
                        <td><h:inputSecret id="password-wk" value='#{workerAdminBean.userPassword}'/></td>
                </tr>
                <tr>
                        <td><h:outputLabel for="firstname-wk"> Име : </h:outputLabel></td>
                </tr>
                <tr>
                        <td><h:inputText id="firstname-wk" value="#{workerAdminBean.firstName}"/></td>
                </tr>
                <tr>
                        <td><h:outputLabel for="secondname-wk"> Презиме : </h:outputLabel></td>
                </tr>
                <tr>
                        <td><h:inputText id="secondname-wk"  value="#{workerAdminBean.secondName}"/></td>
                </tr>
                <tr>
                        <td><h:outputLabel for="usernotes-wk"> Напомене : </h:outputLabel></td>
                </tr>
                <tr>
                        <td><h:inputTextarea cols='50' rows='20' id="usernotes-wk" value="#{workerAdminBean.userNotes}"/></td>
                </tr>
        </table>
        <br>
        <h:commandButton id="add-button-wk" value="Додавање" action='#{workerAdminBean.add}'/>
        <h:commandButton id="remove-button-wk" value="брисање" action='#{workerAdminBean.remove}'/>
        <h:commandButton id="update-button-data-wk" value="измјена података" action='#{workerAdminBean.updateData}'/>
        <h:commandButton id="update-button-data-pw-wk" value="измјена лозинке" action='#{workerAdminBean.updatePassword}'/><br>
        <h:commandButton id="update-button-data-un-wk" value="измјена корисничког имена" action='#{workerAdminBean.updateUsername}'/>
        <h:commandButton id="select-button-wk" value="избор" action='#{workerAdminBean.select}'/>
        <h:commandButton id="reset-button-wk" value="ресетовање" action='#{workerAdminBean.reset}'/><br>
        <h:commandButton id="degrade-button-wk" value="поништавање привилегије" action='#{workerAdminBean.revokeRole}'/><br><br>
    </h:form>
</f:view>
```

master EW-008.STUD-ETF-IP.046_AeroflotUsers / WebContent / WEB-INF / **faces-config.xml**

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

<faces-config
    xmlns="http://java.sun.com/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
    version="2.0">
</faces-config>
```

master EW-008.STUD-ETF-IP.046_AeroflotUsers / src-adminer / studiranje / ip / admin / jsf / bean / **WorkerBean.java**

```java
/**
 * Администраторско зрно за баратање запосленим радницима.
 * Администратор може избрисати, измјенити или додати запосленог.
 * @author VM
 * @version 1.0
 */
@ManagedBean(name = "workerAdminBean", eager = true)
@SessionScoped
public class WorkerBean implements Serializable{
        private static final long serialVersionUID = -2797438451049295859L;

        private String firstName  = "";
        private String secondName = "";
        private String userName = "";
        private String userNotes = "";
        private String userPassword = "";
        private Worker current;
        private HttpSession session;
```

```
51          public void initializeSession(HttpSession session) {
52                  if(this.session!=null) return;
53                  this.session = session;
54          }
55          public HttpSession getSession() {
56                  return session;
57          }
58          public void setFirstName(String firstName) {
59                  if(firstName==null) firstName = "";
60                  this.firstName = firstName;
61          }
62          public void setSecondName(String secondName) {
63                  if(secondName==null) secondName = "";
64                  this.secondName = secondName;
65          }
66          public void setUserName(String userName) {
67                  if(userName==null) userName = "";
68                  this.userName = userName;
69          }
70          public void setUserNotes(String userNotes) {
71                  if(userNotes==null) userNotes = "";
72                  this.userNotes = userNotes;
73          }
74          public String getFirstName() {
75                  return firstName;
76          }
77          public String getSecondName() {
78                  return secondName;
79          }
80          public String getUserName() {
81                  return userName;
82          }
83          public String getUserNotes() {
84                  return userNotes;
85          }
86          public String getUserPassword() {
87                  return userPassword;
88          }
89          public void setUserPassword(String userPassword) {
90                  if(userPassword==null) userPassword="";
91                  this.userPassword = userPassword;
92          }
```

```
94          public List<Worker> list(){
95                  try {
96                          ArrayList<Worker> workers = new ArrayList<>();
97                          Page page = new Page();
98                          page.setPageSize(AppLogicCenter.appLogicCenter.getEngine().getDataSource().count());
99                          for(var data : AppLogicCenter.appLogicCenter.getEngine().getDataSource().list(page)) {
100                                 UserRoleDao roleDAO = DBRoleDAOListener.getUserRoleDAO();
101                                 if(roleDAO.getWorkerRole(data.getCredentials().getUser().getUsername())==null) continue;
102                                 if(!roleDAO.getWorkerRole(data.getCredentials().getUser().getUsername()).getValue().contentEquals("true"))
103                                         continue;
104                                 Worker w = new Worker();
105                                 UserDataDao dataDAO = DBRoleDAOListener.getUserDataDAO();
106                                 UserDataDto dataDTO = dataDAO.get(data.getCredentials().getUser().getUsername());
107                                 if(dataDTO==null) dataDTO = new UserDataDto();
108                                 w.setFirstName(dataDTO.getFirstname());
109                                 w.setSecondName(dataDTO.getSecondname());
110                                 w.setPassword("");
111                                 w.setUserNotes(dataDTO.getUsernote());
112                                 w.setUserName(data.getCredentials().getUser().getUsername());
113                                 workers.add(w);
114                         }
115                         return workers;
116                 }catch(Exception ex) {
117                         return new ArrayList<>();
118                 }
119         }
120
121         public String url(String text) {
122                 try {
123                         return URLEncoder.encode(text, "UTF-8");
124                 }catch(Exception ex) {
125                         return "";
126                 }
127         }
128
129         public Worker current() {
130                 return current;
131         }
132
133         public boolean isSelected() {
134                 return current!=null;
135         }
```

```
137         public void set(String username) {
138                 try {
139                         if(username==null) return;
140                         UserDataDao dataDAO = DBRoleDAOListener.getUserDataDAO();
141                         UserDataDto dataDTO = dataDAO.get(username);
142                         if(dataDTO==null) dataDTO = new UserDataDto();
143                         userName    = username;
144                         firstName   = dataDTO.getFirstname();
145                         secondName = dataDTO.getSecondname();
146                         userNotes    = dataDTO.getUsernote();
147                         userPassword = "";
148                 }catch(Exception ex) {
149                         reset();
150                 }
151         }
152
153         public void reset() {
154                 firstName  = "";
155                 secondName = "";
156                 userName = "";
157                 userNotes = "";
158                 userPassword = "";
159                 current = null;
160                 session = null;
161         }
162
163         public void select(String username) {
164                 try {
165                         if(username==null) return;
166                         UserDataDao dataDAO = DBRoleDAOListener.getUserDataDAO();
167                         UserDataDto dataDTO = dataDAO.get(username);
168                         if(dataDTO==null) dataDTO = new UserDataDto();
169                         Worker user = new Worker();
170                         user.setUserName(dataDTO.getUsername());
171                         user.setFirstName(dataDTO.getFirstname());
172                         user.setSecondName(dataDTO.getSecondname());
173                         user.setUserNotes(dataDTO.getUsernote());
174                         user.setPassword(userPassword);
175                         current = user;
176                 }catch(Exception ex) {
177                         reset();
178                 }
179         }
```

```
181         public void select() {
182                 select(userName);
183         }
184 
185         public void unselect() {
186                 current = null;
187         }
188 
189 
190         public Worker getWorker() {
191                 if(userName.trim().length()==0) return null;
192                 Worker worker = new Worker();
193                 worker.setFirstName(firstName);
194                 worker.setSecondName(secondName);
195                 worker.setUserName(userName);
196                 worker.setPassword(userPassword);
197                 worker.setUserNotes(userNotes);
198                 return worker;
199         }
```

```
	public void add() {
		try {
			if(getWorker()==null) return;
			Worker w = getWorker();
			if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(w.getUserName())!=null) return;
			MySQLUserRegisterDTO dto = new MySQLUserRegisterDTO();
			UserCredentials userCredentials = new UserCredentials();
			userCredentials.setPasswordPlain(userPassword);
			User user = new User();
			user.setUsername(userName);
			userCredentials.setUser(user);
			dto.setCredentials(userCredentials);
			UserDataDao dataDAO = DBRoleDAOListener.getUserDataDAO();
			UserRoleDao roleDAO = DBRoleDAOListener.getUserRoleDAO();
			if(dataDAO==null) return;
			if(roleDAO==null) return;
			UserCredentialsController ucc = new UserCredentialsController();
			String testPassword = ucc.setGoodPassword(userPassword);
			String testUsername = ucc.setGoodUsername(userName);
			if(testPassword==null || testPassword.trim().length()==0) return;
			if(testUsername==null || testUsername.trim().length()==0) return;
			if(!AppLogicCenter.appLogicCenter.getController().register(dto.getCredentials().getUser().getUsername(), userPassword)) return;
			UserDataDto dDto = new UserDataDto();
			dDto.setUsername(userName);
			dDto.setFirstname(firstName);
			dDto.setSecondname(secondName);
			dDto.setUsernote(userNotes);
			dataDAO.insert(dDto);
			dataDAO.bind(userName);
			UserRoleDto rDto = new UserRoleDto();
			rDto.setApplication("aeroflot_users");
			rDto.setKey("worker");
			rDto.setValue("true");
			rDto.setUsername(userName);
			roleDAO.insert(rDto);
			rDto.setKey("administrator");
			rDto.setValue("false");
			roleDAO.insert(rDto);
			rDto.setKey("user");
			rDto.setValue("false");
			roleDAO.insert(rDto);
			userPassword = "";
		}catch(Exception ex) {
			return;
		}
	}
```

```
248         public void remove() {
249                 if(getWorker()==null) return;
250                 try {
251                         LoginBean sessionState = SessionBeansGenerator.loginBean(session);
252                         UserRoleDao roleDAO = DBRoleDAOListener.getUserRoleDAO();
253                         UserRoleDto roleDTO = roleDAO.getWorkerRole(userName);
254                         if(roleDTO == null) return;
255                         if(!roleDTO.getValue().contentEquals("true")) return;
256                         String sessionId = sessionState.getSessionId();
257                         if(sessionState.getUsername().contentEquals(userName)) {
258                                 BaseBean sessionBean = SessionBeansGenerator.baseBean(session);
259                                 sessionBean.getLoginBean().logoutAll();
260                         }
261                         sessionState.setSessionId(sessionId);
262                         roleDAO.deleteRole(userName);
263                         AppLogicCenter.appLogicCenter.getEngine().getDataSource().remove(userName);
264                         reset();
265                 }catch(Exception ex) {
266                         return;
267                 }
268         }
269
270         public void revokeRole() {
271             try {
272                         UserRoleDao workerDAO = DBRoleDAOListener.getUserRoleDAO();
273                         if(workerDAO!=null) {
274                                 UserRoleDto dto = new UserRoleDto();
275                                 dto.setApplication("aeroflot_users");
276                                 dto.setKey("worker");
277                                 dto.setValue("false");
278                                 dto.setUsername(userName);
279                                 workerDAO.putUserRole(dto);
280                         }
281                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
282                         messageBean.setType(MessageType.SUCCESS);
283                         messageBean.setException(null);
284                         messageBean.setMessage("Повлачење привилегије запосленог за "+userName+" је успјешно.");
285                 }catch(Exception ex) {
286                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
287                         messageBean.setType(MessageType.ERROR);
288                         messageBean.setException(ex);
289                         messageBean.setMessage("Повлачење привилегије запосленог за "+userName+" није успјешно.");
290                 }
291         }
```

```
293         public void updateData() {
294                 try {
295                         Worker w = getWorker();
296                         if(w==null) throw new RuntimeException("Worker not found.");
297                         if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(w.getUserName())==null)
298                                 throw new RuntimeException("Worker not found.");
299                         UserRoleDao workerDAO = DBRoleDAOListener.getUserRoleDAO();
300                         UserRoleDto workerDTO = workerDAO.getWorkerRole(w.getUserName());
301                         if(workerDTO==null) throw new RuntimeException("No worker privileges.");
302                         if(!workerDTO.getValue().contentEquals("true")) throw new RuntimeException("No worker privileges.");
303 
304                         UserDataDao wInfoDao = DBRoleDAOListener.getUserDataDAO();
305                         UserDataDto wInfoDto = new UserDataDto();
306                         wInfoDto.setUsername(w.getUserName());
307                         wInfoDto.setFirstname(w.getFirstName());
308                         wInfoDto.setSecondname(w.getSecondName());
309                         wInfoDto.setUsernote(w.getUserNotes());
310                         wInfoDao.put(wInfoDto);
311 
312                         current = null;
313                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
314                         messageBean.setType(MessageType.SUCCESS);
315                         messageBean.setException(null);
316                         messageBean.setMessage("Измјена основних података за запосленог "+userName+" је успјешна.");
317                 }catch(Exception ex) {
318                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
319                         messageBean.setType(MessageType.ERROR);
320                         messageBean.setException(ex);
321                         messageBean.setMessage("Измјена основних података за запосленог "+userName+" није успјешна.");
322                 }
323         }
```

```
379         public void updatePasswordWorker() {
380                 try {
381                         Worker w = current;
382                         if(w==null) throw new RuntimeException("Worker not found.");
383                         if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(w.getUserName())==null)
384                                 throw new RuntimeException("Worker not found.");
385                         UserRoleDao workerDAO = DBRoleDAOListener.getUserRoleDAO();
386                         UserRoleDto workerDTO = workerDAO.getWorkerRole(w.getUserName());
387                         if(workerDTO==null) throw new RuntimeException("No worker privileges.");
388                         if(!workerDTO.getValue().contentEquals("true")) throw new RuntimeException("No worker privileges.");
389                         UserCredentialsController ucc = new UserCredentialsController();
390                         Worker neo = getWorker();
391                         if(neo==null) throw new RuntimeException("Worker not found");
392                         if(!neo.getUserName().contentEquals(w.getUserName())) throw new RuntimeException("Update workers username not synchronized.");
393 
394                         String testPassword = ucc.setGoodPassword(w.getPassword());
395                         if(testPassword==null) throw new RuntimeException("Worker. Password not good.");
396                         if(testPassword.trim().length()==0) throw new RuntimeException("Worker. Password not good.");
397 
398                         try {
399                                 if(!AppLogicCenter.appLogicCenter.getController().updatePassword(w.getUserName(), w.getPassword(), neo.getPassword()))
400                                         throw new RuntimeException();
401                         }catch(Exception ex) {
402                                 throw new RuntimeException("Worker current password not correct.");
403                         }
404 
405                         current = null;
406 
407                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
408                         messageBean.setType(MessageType.SUCCESS);
409                         messageBean.setException(null);
410                         messageBean.setMessage("Измјена лозинке за запосленог "+userName+" је успјешна.");
411                 }catch(Exception ex) {
412                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
413                         messageBean.setType(MessageType.ERROR);
414                         messageBean.setException(ex);
415                         messageBean.setMessage("Измјена лозинке за запосленог "+userName+" није успјешна.");
416                 }
417         }
```

```
325         public void updateUsername() {
326                 try {
327                         Worker w = current;
328                         if(w==null) throw new RuntimeException("Worker not found.");
329                         if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(w.getUserName())==null)
330                                 throw new RuntimeException("Worker not found.");
331                         UserRoleDao workerDAO = DBRoleDAOListener.getUserRoleDAO();
332                         UserRoleDto workerDTO = workerDAO.getWorkerRole(w.getUserName());
333                         if(workerDTO==null) throw new RuntimeException("No worker privileges.");
334                         if(!workerDTO.getValue().contentEquals("true")) throw new RuntimeException("No worker privileges.");
335
336                         UserCredentialsController ucc = new UserCredentialsController();
337                         Worker neo = getWorker();
338
339                         if(neo==null) throw new RuntimeException("New worker data not found");
340                         if(!neo.getUserName().contentEquals(w.getUserName())) {
341                                 if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(neo.getUserName())!=null)
342                                         throw new RuntimeException("New worker username already exists. Duplication is error.");
343
344                                 String testUsername = ucc.setGoodUsername(neo.getUserName());
345                                 if(testUsername==null) throw new RuntimeException("New worker. Username not good.");
346                                 if(testUsername.trim().length()==0) throw new RuntimeException("New worker. Username not good.");
347
348                                 try {
349                                         BaseBean baseBean = SessionBeansGenerator.baseBean(session);
350                                         LoginBean sessionState = SessionBeansGenerator.loginBean(session);
351                                         List<String> sessionList = new ArrayList<>();
352                                         if(w.getUserName().contentEquals(sessionState.getUsername())) sessionState.logoutAll();
353                                         for(Session s: baseBean.getSessionListBean().listForUser(w.getUserName()))
354                                                 sessionList.add(s.getSessionId());
355                                         sessionState.logout(sessionList);
356
357                                         if(!AppLogicCenter.appLogicCenter.getController().getDataSource().updateUsername(w.getUserName(), neo.getUserName()))
358                                                 throw new RuntimeException();
359                                 }catch(Exception ex) {
360                                         throw new RuntimeException("Worker update username not successfull.");
361                                 }
362                         }
363
364
365                         current = null;
366
367                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
368                         messageBean.setType(MessageType.SUCCESS);
369                         messageBean.setException(null);
370                         messageBean.setMessage("Измјена корисничког имена за запосленог "+userName+" је успјешна.");
371                 }catch(Exception ex) {
372                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
373                         messageBean.setType(MessageType.ERROR);
374                         messageBean.setException(ex);
375                         messageBean.setMessage("Измјена корисничког имена за запосленог "+userName+" није успјешна.");
```

```
419         public void updatePassword() {
420                 try {
421                         Worker w = getWorker();
422                         if(w==null) throw new RuntimeException("Worker not found.");
423                         if(AppLogicCenter.appLogicCenter.getEngine().getDataSource().get(w.getUserName())==null)
424                                 throw new RuntimeException("Worker not found.");
425                         UserRoleDao workerDAO = DBRoleDAOListener.getUserRoleDAO();
426                         UserRoleDto workerDTO = workerDAO.getWorkerRole(w.getUserName());
427                         if(workerDTO==null) throw new RuntimeException("No worker privileges.");
428                         if(!workerDTO.getValue().contentEquals("true")) throw new RuntimeException("No worker privileges.");
429                         UserCredentialsController ucc = new UserCredentialsController();
430 
431                         String testPassword = ucc.setGoodPassword(w.getPassword());
432                         if(testPassword==null) throw new RuntimeException("Worker. Password not good.");
433                         if(testPassword.trim().length()==0) throw new RuntimeException("Worker. Password not good.");
434 
435                         try {
436                                 if(!AppLogicCenter.appLogicCenter.getController().getDataSource().updatePassword(w.getUserName(), w.getPassword()))
437                                         throw new RuntimeException();
438                         }catch(Exception ex) {
439                                 throw new RuntimeException("Worker update password not successfull.");
440                         }
441 
442                         current = null;
443 
444                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
445                         messageBean.setType(MessageType.SUCCESS);
446                         messageBean.setException(null);
447                         messageBean.setMessage("Измјена лозинке за запосленог "+userName+" је успјешна.");
448                 }catch(Exception ex) {
449                         MessageBean messageBean = DesignBeansGenerator.messageBean(session);
450                         messageBean.setType(MessageType.ERROR);
451                         messageBean.setException(ex);
452                         messageBean.setMessage("Измјена лозинке за запосленог "+userName+" није успјешна.");
453                 }
454         }
455 }
```