```
 1 import java.io.*;
 2 import java.sql.*;
 3 import javax.servlet.*;
 4 import javax.servlet.http.*;
 5 import java.util.Calendar;
 6 import java.text.DateFormat;
 7 import java.util.Date;
 8
 9
10 public class Kalenteri extends HttpServlet {
11
12   final String dbDriver="org.postgresql.Driver";
13   final String dbServer ="jdbc:postgresql://localhost:10388/tsoha";
14   final String dbUser= "avtanska";        // replace with your db user account
15   final String dbPassword ="postgres"; // replace with your password
16
17
18   public void service(HttpServletRequest req, HttpServletResponse res)
19          throws ServletException, IOException {
20
21     ServletOutputStream out;
22     res.setContentType("text/html");
23     out= res.getOutputStream();
24
25     Calendar kalenteri = Calendar.getInstance();
26
27     HttpSession session = req.getSession(false);
28     String sessionTunnus = (String)session.getValue("sessionTunnus");
29
30
31     /*
32      * Tämä muuttuja määrittelee kuinka monta pikseliä yhden
33      * tunnin korkeus kalenterissa on
34      */
35
36     final int TUNNIN_KORKEUS = 26;
37
38
39     String[] valitutTaulu = req.getParameterValues("valitut");
40     String poista = req.getParameter("poista");
41     String henkilo = req.getParameter("henkilo");
42     String ryhma = req.getParameter("ryhma");
43     String alkupvm = req.getParameter("alkupvm");
44     String viikkoAlusta = req.getParameter("viikkoAlusta");
45     String valitut = "";
46     String henkilöTaiRyhmä = "";
47
48
49     if (henkilo != null && henkilo.equals("tyhjä")) {
50       henkilo = null;
51     }
52     if (ryhma != null && ryhma.equals("tyhjä")) {
53       ryhma = null;
54     }
55     if (viikkoAlusta == null) {
56       viikkoAlusta = "";
57     }
58
59     out.println("<html><head><title>Database query from DB (tsoha)</title>" +
60     "<link rel='stylesheet' style='text/css' " +
61     "href='http://db.cs.helsinki.fi/u/avtanska/tsoha/perus.css'></head>" +
62     "<body>");
63
64
65
66     /*
67      * Tarkistetaan onko valittu ryhmää tai henkilöä.
68      */
69
```

```
70      if (ryhma == null && henkilo == null) {
71        out.println("<h3>Valitse ryhmä tai henkilö.</h3></body></html>");
72      }
73
74
75      /*
76       * Jos valinnat kunnossa aloitetaan kalenterin generointi
77       */
78
79      else {
80
81        Connection con=null;
82        con= createDbConnection(dbDriver,dbServer,dbUser,dbPassword,out);
83        if (con==null) {
84           out.println("</body></html>");
85           return;
86         }
87
88        Statement stmt = null;
89        ResultSet rs = null;
90
91
92        /*
93         * Tulostetaan sivun otsikko ja henkilön kalenterin tapauksessa
94         * henkilön tiedot ja ryhmän tapauksessa ryhmän jäsenet
95         */
96
97        if (henkilo != null) {
98          henkilöTaiRyhmä = "henkilo=" + henkilo;
99          out.println("<h3>Varaukset henkilölle '" + henkilo + "'</h3>");
100
101         try {
102           stmt = con.createStatement();
103           rs = stmt.executeQuery("SELECT * FROM henkilö WHERE tunnus='" +
104                                  henkilo + "'");
105
106           while (rs.next()) {
107             out.println("<p><b>Nimi:</b> " + rs.getString("etunimi") +
108                         " " + rs.getString("sukunimi") + "<br />");
109             out.println("<b>Puhelin:</b> " + rs.getString("puh") +
110                         "<br />");
111             out.println("<b>Email:</b> " + rs.getString("sposti") +
112                         "</p>");
113           }
114
115         } catch (Exception e) {
116            out.println("Tietokantavirhe: " + e);
117         }
118
119
120       }
121       else if (ryhma != null) {
122         henkilöTaiRyhmä = "ryhma=" + ryhma;
123         out.println("<h3>Varaukset ryhmälle '" + ryhma + "'</h3>");
124
125         try {
126           stmt = con.createStatement();
127           rs = stmt.executeQuery("SELECT tunnus FROM jäsenyys WHERE ryhmä='" +
128                                  ryhma + "'");
129
130           out.println("<b>Jäsenet:</b> ");
131
132           while (rs.next()) {
133             if (!rs.isLast()) {
134               out.println(rs.getString("tunnus") + ", ");
135             }
136             else {
137               out.println(rs.getString("tunnus"));
138             }
```

```
139
140           }
141
142         } catch (Exception e) {
143            out.println("Tietokantavirhe: " + e);
144         }
145       }
146
147
148       out.println("<form method='post' " +
149          "action='http://db.cs.helsinki.fi/s/avtanska/Kalenteri'>");
150
151
152       /*
153        * Jos painettu 'Poista valitut'-nappia, muunnetaan valinnat
154        * SQL-lauseeseen sopivaan muotoon
155        */
156
157       if (poista != null && valitutTaulu != null) {
158         for (int i = 0; i < valitutTaulu.length; i++) {
159           valitut += valitutTaulu[i] + ",";
160         }
161         valitut = valitut.substring(0, valitut.length()-1);
162       }
163
164
165       /*
166        * Alustetaan kalenteri osoittamaan käynnissä olevan
167        * viikon ensimmäiseen päivään (Euroopassa maanantai)
168        */
169
170       DateFormat df = DateFormat.getDateInstance();
171       Date tempPvm = null;
172       int alkuun = 0;
173
174       if (alkupvm == null) {
175         alkuun =
176            kalenteri.getFirstDayOfWeek() - kalenteri.get(Calendar.DAY_OF_WEEK);
177         if (kalenteri.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY) {
178           alkuun = -6;
179         }
180         kalenteri.add(Calendar.DATE, alkuun);
181         alkupvm = "" + df.format(kalenteri.getTime());
182       }
183
184
185       /*
186        * Jos kalenteriin tullaan koko vuoden kalenterin kautta,
187        * asetetaan kalenteri osoittamaan sieltä valitun päivän
188        * sisältämän viikon alkuun.
189        */
190
191       if (viikkoAlusta.equals("true")) {
192         try {
193           tempPvm = df.parse(alkupvm);
194         } catch (Exception e) {
195           out.println("Virhe: " + e);
196         }
197
198         kalenteri.setTime(tempPvm);
199
200         alkuun =
201            kalenteri.getFirstDayOfWeek() - kalenteri.get(Calendar.DAY_OF_WEEK);
202         if (kalenteri.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY) {
203           alkuun = -6;
204         }
205         kalenteri.add(Calendar.DATE, alkuun);
206         alkupvm = "" + df.format(kalenteri.getTime());
207       }
```

```
208
209       try {
210         tempPvm = df.parse(alkupvm);
211       } catch (Exception e) {
212         out.println("Virhe: " + e);
213       }
214
215       kalenteri.setTime(tempPvm);
216
217
218       /*
219        * Tässä pyöritellään kalenteria edestakaisin, jotta saadaan
220        * oikeat päivämäärät sen tulostusta ohjaaville nuolille, sekä
221        * päivämäärät tulostettavan aikavälin päätepisteille
222        */
223
224       kalenteri.add(Calendar.DATE, -1);
225       String taakse = df.format(kalenteri.getTime());
226
227       kalenteri.add(Calendar.DATE, 2);
228       String eteen = df.format(kalenteri.getTime());
229
230       kalenteri.add(Calendar.DATE, -8);
231       String viikkoTaakse = df.format(kalenteri.getTime());
232
233       kalenteri.add(Calendar.DATE, 14);
234       String viikkoEteen = df.format(kalenteri.getTime());
235
236       kalenteri.add(Calendar.DATE, -7);
237       String viikonAlku = CheckDate.c(kalenteri.get(Calendar.DATE)) +
238                           "." + CheckDate.c(kalenteri.get(Calendar.MONTH)+1) +
239                           "." + kalenteri.get(Calendar.YEAR);
240
241       out.println("<table width='800' class='blank'><tr><td>" +
242                   "Varaukset ajalle: <b>" + viikonAlku + " - ");
243
244       kalenteri.add(Calendar.DATE, 6);
245       String viikonLoppu = CheckDate.c(kalenteri.get(Calendar.DATE)) +
246                            "." + CheckDate.c(kalenteri.get(Calendar.MONTH)+1) +
247                            "." + kalenteri.get(Calendar.YEAR);
248
249       out.println(viikonLoppu + "</b></td>");
250
251       kalenteri.add(Calendar.DATE, -6);
252
253
254       String vuosiNyt = "" + kalenteri.get(Calendar.YEAR);
255
256       out.println("<td><a href='http://db.cs.helsinki.fi/" +
257                   "s/avtanska/VuosiKalenteri?henkilo=" + henkilo  +
258                   "&ryhma=" + ryhma + "&vuosi=" + vuosiNyt + "'>" +
259                   "Koko vuoden kalenteri</a></td>");
260
261
262       /*
263        * Varausten poistoon tarkoitetut napit ja muutama
264        * piilokenttä, jotta tarvittavat tiedot pysyvät
265        * mukana matkassa
266        */
267
268       out.println("<td><input type='submit' name='poista' " +
269                   "value='Poista valitut'><input type='reset' " +
270                   "name='reset' value='Tyhjennä valinnat'>");
271
272       if (henkilo != null) {
273         out.println("<input type='hidden' name='henkilo' value='" +
274                     henkilo + "'>");
275       }
276       else if (ryhma != null) {
```

```
277         out.println("<input type='hidden' name='ryhma' value='" +
278                     ryhma + "'>");
279       }
280
281       out.println("<input type='hidden' name='alkupvm' value='" +
282                   viikonAlku + "'></td></tr></table>");
283
284
285
286       /*
287        * Kalenterin tulostus alkaa
288        */
289
290       out.println("<table cellpadding='0' cellspacing='1' " +
291                   "style='background: #000000'>");
292
293
294       out.println("<tr><td class='kalenteri' style='padding: 4 0 4 0' " +
295                   "colspan='8' align='center'>");
296
297       /*
298        * Ohjausnuolet
299        */
300
301       out.println("<a href='?alkupvm=" + viikkoTaakse + "&" + henkilöTaiRyhmä +
302                   "'>&lt;&lt;</a> ");
303       out.println("  <a href='?alkupvm=" + taakse + "&" +
304                   henkilöTaiRyhmä + "'>&lt;</a> ");
305       out.println(" <a href='?alkupvm=" + eteen + "&" + henkilöTaiRyhmä +
306                   "'>&gt;</a> ");
307       out.println("  <a href='?alkupvm=" + viikkoEteen + "&" +
308                   henkilöTaiRyhmä + "'>&gt;&gt;</a></td></tr><tr>");
309
310
311       /*
312        * Päivien otsikot kalenteriin
313        */
314
315       out.println("<th>klo</th>");
316
317       for (int i = 0; i < 7; i++) {
318         int viikonPäivä = kalenteri.get(Calendar.DAY_OF_WEEK);
319
320         switch (viikonPäivä) {
321           case Calendar.MONDAY:
322              out.println("<th width='100'>Maanantai</th>"); break;
323           case Calendar.TUESDAY:
324              out.println("<th width='100'>Tiistai</th>"); break;
325           case Calendar.WEDNESDAY:
326              out.println("<th width='100'>Keskiviikko</th>"); break;
327           case Calendar.THURSDAY:
328              out.println("<th width='100'>Torstai</th>"); break;
329           case Calendar.FRIDAY:
330              out.println("<th width='100'>Perjantai</th>"); break;
331           case Calendar.SATURDAY:
332              out.println("<th width='100'>Lauantai</th>"); break;
333           case Calendar.SUNDAY:
334              out.println("<th width='100'>Sunnuntai</th>"); break;
335         }
336
337         kalenteri.add(Calendar.DATE, 1);
338       }
339
340       kalenteri.add(Calendar.DATE, -7);
341
342       out.println("</tr>");
343
344
345
```

```
346         /*
347          * Varauksien käsittely alkaa
348          */
349
350         con=null;
351         con= createDbConnection(dbDriver,dbServer,dbUser,dbPassword,out);
352         if (con==null) {
353            out.println("</body></html>");
354            return;
355         }
356
357         int varaustenMäärä = 0;
358         int aikaisinVaraus = 0;
359
360
361         stmt = null;
362         rs = null;
363
364         try {
365           stmt = con.createStatement();
366
367
368           /*
369            *  Haetaan viikon aikaisimman varauksen alkuaika, jotta voidaan
370            *  halutessa trimmata turhat tyhjät pois aamuista kalenterissa
371            */
372
373           if (henkilo != null) {
374             rs = stmt.executeQuery(
375                 "select min(alkuaika) as aikaisin from varaus where tunnus='" +
376                 henkilo + "' and pvm between to_date('" + viikonAlku +
377                 "', 'DD.MM.YYYY') and to_date('" + viikonLoppu +
378                 "', 'DD.MM.YYYY')");
379           }
380           else if (ryhma != null) {
381             rs = stmt.executeQuery(
382                 "select min(alkuaika) as aikaisin from varaus where ryhmä='" +
383                 ryhma + "' and pvm between to_date('" + viikonAlku +
384                 "', 'DD.MM.YYYY') and to_date('" + viikonLoppu +
385                 "', 'DD.MM.YYYY')");
386           }
387
388
389           if (rs.next()) {
390             if (rs.getString("aikaisin") != null) {
391               aikaisinVaraus =
392                  Integer.parseInt((rs.getString("aikaisin")).substring(0,2));
393             }
394           }
395
396
397           /*
398            *  Kommentoi 'aikaisinVaraus = 0' rivi pois, jos
399            *  haluat, että tila ennen viikon aikaisinta
400            *  varausta trimmataan pois. Nyt ominaisuus on
401            *  käytössä vain admin-käyttäjällä testitarkoituksessa.
402            */
403
404            if (!sessionTunnus.equals("admin")) {
405              aikaisinVaraus = 0;
406            }
407
408
409
410           /*
411            *  Varausten poisto, ryhmän varauksen poistaminen poistaa
412            *  varauksen myös kaikilta ryhmän jäseniltä
413            */
414
```

```
415         if (valitutTaulu != null && poista != null) {
416
417           if (ryhma != null) {
418
419
420             /*
421              * Otetaan talteen poistettavien ryhmävarausten tiedot
422              */
423
424             rs = stmt.executeQuery("SELECT * FROM varaus WHERE vid IN " +
425                 "(" + valitut + ")");
426
427
428             /*
429              * Poistetaan ryhmän jäseniltä varaukset, joiden tiedot
430              * täsmäävät ylläsaatujen kanssa
431              */
432
433             while (rs.next()) {
434                String deleteJäseneltä =
435                  "DELETE FROM varaus WHERE tunnus IN " +
436                  "(SELECT tunnus FROM jäsenyys WHERE ryhmä='" +
437                  ryhma + "') AND pvm='" + rs.getString("pvm") + "' AND " +
438                  "alkuaika='" + rs.getString("alkuaika") + "' AND " +
439                  "kesto='" + rs.getString("kesto") + "' AND " +
440                   "aihe='" + rs.getString("aihe") + "' AND " +
441                  "näkyvyys='" + rs.getString("näkyvyys") + "' AND " +
442                  "varaaja='" + rs.getString("varaaja") + "'";
443
444                int count = stmt.executeUpdate(deleteJäseneltä);
445             }
446           }
447
448
449           /*
450            * Yleinen poisto, eli poistaa vain ja ainoastaan varaukset
451            * jotka käyttäjä on ruksannut
452            */
453
454           String queryDelete = "DELETE FROM varaus WHERE vid IN " +
455                                "(" + valitut + ")";
456
457           int count = stmt.executeUpdate(queryDelete);
458
459
460         } // loppu if poistetaan varauksia
461
462
463       } catch (SQLException ee) {
464             out.println("Tietokantavirhe "+ee.getMessage());
465       } finally {
466           try {
467             if (rs!=null) rs.close();
468             if (stmt!=null) stmt.close();
469
470           } catch(SQLException e) {
471             out.println("An SQL Exception was thrown.");
472           }
473       }
474
475       out.println("<tr><td class='kalenteri' valign='top'>" +
476                   "<table class='ajat' cellpadding='0' cellspacing='0'>");
477
478       for (int i = (0 + aikaisinVaraus); i < 24; i++) {
479         if (i == 23) {
480           out.println("<tr><td class='kalenteri' valign='top' height='" +
481                       ((TUNNIN_KORKEUS)-1) + "'>" + CheckDate.c(i) +
482                       ":00</td></tr>");
483         }
```

```
484         else {
485           out.println("<tr><td class='kalenteri' valign='top' height='" +
486                       TUNNIN_KORKEUS + "'>" + CheckDate.c(i) + ":00</td></tr>");
487         }
488       }
489
490       out.println("</table></td>");
491
492
493
494       /*
495        * Käydään seitsemän päivää kalenterin tämänhetkisestä päivämäärästä
496        * lähtien läpi
497        */
498
499       for (int i = 0; i < 7; i++) {  // forin alku
500
501         out.println("<td class='kalenteri' style='background-image: " +
502             "url(http://db.cs.helsinki.fi/u/avtanska/tsoha/img/kale_bg.gif)'" +
503             " valign='top'>");
504
505         String käsiteltäväPäivä =
506             "" + CheckDate.c(kalenteri.get(Calendar.DATE)) + "." +
507             CheckDate.c((kalenteri.get(Calendar.MONTH)+1)) + "." +
508             kalenteri.get(Calendar.YEAR);
509
510
511         try {
512           stmt = con.createStatement();
513           double edellisenLoppu = 0;
514           boolean päivänEnsimmäinen = true;
515
516
517           /*
518            *  Haetaan joko henkilön tai ryhmän varaukset
519            */
520
521           if (henkilo != null) {
522             rs = stmt.executeQuery(
523                 "SELECT * FROM varaus WHERE tunnus='" + henkilo +
524                 "' AND pvm=to_date('" + käsiteltäväPäivä +
525                 "', 'DD.MM.YYYY') ORDER BY alkuaika");
526           }
527           else if (ryhma != null) {
528             rs = stmt.executeQuery(
529                 "SELECT * FROM varaus WHERE ryhmä='" + ryhma +
530                 "' and pvm=to_date('" + käsiteltäväPäivä +
531                 "', 'DD.MM.YYYY') order by alkuaika");
532           }
533
534
535
536           /*
537            *  Käydään yhden päivän kaikki varaukset läpi yksi kerrallaan
538            */
539
540           boolean onkoVarauksia = false;
541
542           while(rs.next()) {
543
544             onkoVarauksia = true;
545
546             Calendar varausKesto = Calendar.getInstance();
547             String poistoNappi = "";
548             String varausPvm = rs.getString("pvm");
549             String alkuAikaString = rs.getString("alkuaika");
550             String varausAlkaa = "";
551             String varausLoppuu = "";
552             double kesto = 0;
```

```
553             double tyhjaTila = 0;
554             double alkuAika = 0;
555
556             alkuAika = Integer.parseInt(alkuAikaString.substring(0,2));
557             kesto = (new Double(rs.getString("kesto"))).doubleValue();
558
559             if (((rs.getString("alkuaika")).substring(3,5)).equals("30")) {
560               alkuAika = alkuAika + .5;
561             }
562
563
564
565
566             /*
567              *  Lasketaan ja tulostellaan tyhjä tila ennen
568              *  varausta
569              */
570
571             if (päivänEnsimmäinen) {
572               tyhjaTila = alkuAika - aikaisinVaraus;
573             }
574             else {
575               tyhjaTila = alkuAika - edellisenLoppu;
576             }
577
578             edellisenLoppu = alkuAika + kesto;
579
580
581             out.println("<table cellspacing='0' cellpadding='0' " +
582                         "width='100' class='varaus'>");
583
584
585             if (päivänEnsimmäinen && tyhjaTila != 0) {
586               out.println("<tr><td width='100' class='kalenteri' " +
587                  "style='background: url(http://db.cs.helsinki.fi/u/avtanska" +
588                  "/tsoha/img/transparent.gif)' height='" +
589                  ((tyhjaTila*TUNNIN_KORKEUS)-1) + "'></td></tr>");
590             } else if (tyhjaTila != 0) {
591               out.println("<tr><td width='100' class='kalenteri'  " +
592                  "style='background: url(http://db.cs.helsinki.fi/u/avtanska" +
593                  "/tsoha/img/transparent.gif)' height='" +
594                  ((tyhjaTila*TUNNIN_KORKEUS)-1) + "'></td></tr>");
595             }
596
597
598             /*
599              * Jos varaus ei ala kello 00:00, tulostetaan varauksen
600              * yläreunaan musta viiva
601              */
602
603             if ((päivänEnsimmäinen && tyhjaTila != 0) || tyhjaTila != 0) {
604               out.println("<tr><td height='1' style='background: #000000'>" +
605                           "</td></tr>");
606             }
607
608
609             /*
610              * Sitten itse varauksen tulostus
611              */
612
613             out.println("<tr><td valign='top' class='varaus'" +
614                         " height='" + ((kesto*TUNNIN_KORKEUS)-1) + "'>");
615
616
617             if (sessionTunnus.equals(rs.getString("varaaja")) ||
618                 sessionTunnus.equals("admin")) {
619               poistoNappi = "<input type='checkbox' name='valitut' " +
620                             "value='" + rs.getString("vid") + "'>";
621             }
```

```
622
623
624             /*
625              *  Valmistellaan varauksen alku-, ja loppuaika
626              *  tulostuskuntoon
627              */
628
629             varausAlkaa = (rs.getString("alkuaika")).substring(0,5);
630             varausLoppuu = "";
631
632             varausKesto.clear();
633             varausKesto.set(Calendar.HOUR,
634                             Integer.parseInt(alkuAikaString.substring(0,2)));
635             varausKesto.set(Calendar.MINUTE,
636                             Integer.parseInt(alkuAikaString.substring(3,5)));
637             varausKesto.add(Calendar.MINUTE, (int)(kesto * 60));
638
639             varausLoppuu +=
640                "" + CheckDate.c(varausKesto.get(Calendar.HOUR_OF_DAY)) + ":";
641
642             varausLoppuu +=
643               "" + CheckDate.c(varausKesto.get(Calendar.MINUTE));
644
645
646             /*
647              *  Tulostetaan varauksen tietoja
648              */
649
650             if (rs.getString("näkyvyys").equals("2") &&
651                 !sessionTunnus.equals(rs.getString("tunnus")) &&
652                 !sessionTunnus.equals(rs.getString("varaaja")) &&
653                 !sessionTunnus.equals("admin")) {
654               out.println("<p class='varaus'><b>Varattu</b><br />" +
655                 varausAlkaa + "-" + varausLoppuu + "</p>");
656             }
657             else {
658               if (kesto < 2) {
659                 out.println("<p class='varaus'><b>" + rs.getString("aihe") +
660                             "</b>  " + poistoNappi + "</p>");
661               }
662               else {
663                 out.println("<p class='varaus'><b>" +  rs.getString("aihe") +
664                             "</b><br />" + varausAlkaa + "-" + varausLoppuu +
665                             "<br />Varaaja: " + rs.getString("varaaja") +
666                             poistoNappi + "</p>");
667               }
668             }
669
670             out.println("</td></tr>");
671
672             /*
673              * Tulostetaan varauksen alareunaan musta viiva, jos varaus
674              * ei lopu kello 24:00
675              */
676
677             if (alkuAika + kesto < 24) {
678               out.println("<tr><td height='1' style='background: #000000'>" +
679                           "</td></tr>");
680             }
681
682             out.println("</table>");
683
684             päivänEnsimmäinen = false;
685             varaustenMäärä++;
686
687           } // loppu while
688
689           if (!onkoVarauksia) {
690             out.println(" ");
```

```
691           }
692
693         } catch (SQLException ee) {
694               out.println("Tietokantavirhe "+ee.getMessage());
695         } finally {
696               try {
697                  if (rs!=null) rs.close();
698                  if (stmt!=null) stmt.close();
699
700               } catch(SQLException e) {
701                  out.println("An SQL Exception was thrown.");
702               }
703         }
704
705         kalenteri.add(Calendar.DATE, 1);
706         out.println("</td>");
707
708       } // loppu for
709
710
711       /*
712        *  Suljetaan yhteys
713        */
714
715       try {
716         if (rs!=null) rs.close();
717         if (stmt!=null) stmt.close();
718         con.close();
719       } catch(SQLException e) {
720           out.println("An SQL Exception was thrown.");
721       }
722
723       out.println("</tr><tr><th colspan='8'>Varausten lukumäärä: " +
724                   varaustenMäärä + "</th></tr></table></form></body></html>");
725
726
727
728
729
730     } // loppu else ryhmä ja henkilö ei null
731
732   }
733
734
735
736   private Connection createDbConnection(
737       String dbDriver, String dbServer, String dbUser, String dbPassword,
738       ServletOutputStream out) throws IOException {
739
740       try{
741           Class.forName(dbDriver);
742       } catch (ClassNotFoundException e) {
743             out.println("Couldn't find driver "+dbDriver);
744             return null;
745       }
746       Connection con=null;
747       try {
748          con = DriverManager.getConnection(dbServer,dbUser,dbPassword);
749       } catch (SQLException se) {
750             out.println("Couldn\'t get connection to "+dbServer+
751                         " for "+ dbUser+"<br>");
752             out.println(se.getMessage());
753       }
754       return con;
755   }
756
757
758 }
```