```
 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 import java.util.Locale;
 9
10 public class RyhmanAjat 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     ServletOutputStream out;
21     res.setContentType("text/html");
22     out= res.getOutputStream();
23
24     Calendar kalenteri = Calendar.getInstance();
25
26     HttpSession session = req.getSession(false);
27     String sessionTunnus = (String)session.getValue("sessionTunnus");
28     String omatryhmat = req.getParameter("omatryhmat");
29
30
31     out.println("<html><head><title>Database query from DB (tsoha)</title>" +
32                 "<link rel='stylesheet' style='text/css' " +
33                 "href='http://db.cs.helsinki.fi/u/avtanska/tsoha/perus.css'>" +
34                 "</head><body bgcolor=white>");
35
36     /*
37      * Tarkistetaan onko valittu ryhmää
38      */
39
40     if (omatryhmat.equals("tyhjä")) {
41       out.println("<h3>Valitse ryhmäsi.</h3>");
42     }
43
44     /*
45      * Jos valinnat kunnossa aloitetaan vapaiden aikojen laskeminen
46      */
47
48     else {
49
50       out.println("<form method='post' action='http://db.cs.helsinki.fi/" +
51                   "s/avtanska/Kalenteri'>");
52
53       String alkupvm = req.getParameter("alkupvm");
54
55
56       /*
57        * Alustetaan kalenteri osoittamaan käynnissä olevan
58        * viikon ensimmäiseen päivään (Euroopassa maanantai)
59        */
60
61       DateFormat df = DateFormat.getDateInstance();
62       Date tempPvm = null;
63       int alkuun = 0;
64
65       if (alkupvm == null) {
66         alkuun =
67            kalenteri.getFirstDayOfWeek() - kalenteri.get(Calendar.DAY_OF_WEEK);
68         if (kalenteri.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY) {
69           alkuun = -6;
```

```
70          }
71          kalenteri.add(Calendar.DATE, alkuun);
72          alkupvm = "" + df.format(kalenteri.getTime());
73        }
74
75        try {
76          tempPvm = df.parse(alkupvm);
77        } catch (Exception e) {
78          out.println("Virhe: " + e);
79        }
80
81        kalenteri.setTime(tempPvm);
82
83
84        /*
85         * Tässä pyöritellään kalenteria edestakaisin, jotta saadaan
86         * oikeat päivämäärät sen tulostusta ohjaaville nuolille, sekä
87         * päivämäärät tulostettavan aikavälin päätepisteille
88         */
89
90        kalenteri.add(Calendar.DATE, -1);
91        String taakse = df.format(kalenteri.getTime());
92
93        kalenteri.add(Calendar.DATE, 2);
94        String eteen = df.format(kalenteri.getTime());
95
96        kalenteri.add(Calendar.DATE, -8);
97        String viikkoTaakse = df.format(kalenteri.getTime());
98
99        kalenteri.add(Calendar.DATE, 14);
100       String viikkoEteen = df.format(kalenteri.getTime());
101
102
103       kalenteri.add(Calendar.DATE, -7);
104
105       String viikonAlku = CheckDate.c(kalenteri.get(Calendar.DATE)) +
106                           "." + CheckDate.c(kalenteri.get(Calendar.MONTH)+1) +
107                           "." + kalenteri.get(Calendar.YEAR);
108
109       out.println("<h3>Ryhmän '" + omatryhmat + "' kaikkien henkilöiden " +
110                   "varaukset koottuna yhteen:</h3>");
111
112
113       /*
114        * Solujen värit selityksineen
115        */
116
117       out.println("<p><table><tr><td style='background: #000000' " +
118                   "width='100' height='13'> </td>" +
119                   "<td style='background: #ffffff' width='30' " +
120                   "align='center'>=</td>" +
121                   "<td style='background: #ffffff'>varattu</td>" +
122                   "</tr></table></p>");
123
124       out.println("<p><table><tr><td width='100' height='15'>" +
125                   "<table cellpadding='0' cellspacing='1' bgcolor='#000000'>" +
126                   "<tr><td width='100' height='15' style='background: " +
127                   "#eeeeee'> </td></tr></table></td>" +
128                   "<td style='background: #ffffff' width='30' "+
129                   "align='center'>=</td>" +
130                   "<td style='background: #ffffff'>vapaa kaikilla</td>" +
131                   "</tr></table></p>");
132
133
134       out.println("<p>Varaukset ajalle: <b>" + viikonAlku + " - ");
135
136       kalenteri.add(Calendar.DATE, 6);
137       String viikonLoppu = CheckDate.c(kalenteri.get(Calendar.DATE)) +
138                            "." + CheckDate.c(kalenteri.get(Calendar.MONTH)+1) +
```

```
139                          "." + kalenteri.get(Calendar.YEAR);
140
141       out.println(viikonLoppu + "</b></p>");
142
143       kalenteri.add(Calendar.DATE, -6);
144
145
146       Connection con=null;
147       con= createDbConnection(dbDriver,dbServer,dbUser,dbPassword,out);
148       if (con==null) {
149          out.println("</body></html>");
150          return;
151       }
152
153
154       /*
155        * Luodaan 48 x 7 taulukko, johon kootaan kaikkien ryhmän
156        * jäsenten viikon varaukset. 48 siksi, että puoli tuntia
157        * on pienin kalenterin tuntema aikayksikkö 24*2 = 48
158        *
159        * Luodaan myös väliaikainen klooni oikeasta kalenterista,
160        * jonka avulla käydään viikon päivät läpi.
161        */
162
163       boolean[][] ryhmänVaraukset = new boolean[48][7];
164       Calendar tempKalenteri = (Calendar)kalenteri.clone();
165
166
167       /*
168        * Alustetaan taulukko. Jos taulukossa arvo true, niin
169        * jollakin ryhmän henkilöllä on varaus kyseisessä kohdassa
170        */
171
172       for (int i = 0; i < ryhmänVaraukset.length; i++) {
173         for (int j = 0; j < ryhmänVaraukset[i].length; j++) {
174           ryhmänVaraukset[i][j] = false;
175         }
176       }
177
178       Statement stmt1 = null;
179       Statement stmt2 = null;
180       ResultSet rs1 = null;
181       ResultSet rs2 = null;
182
183
184       try {
185         stmt1 = con.createStatement();
186         stmt2 = con.createStatement();
187
188
189         /*
190          * Haetaan ryhmän jäsenet tietokannasta
191          */
192
193         String selectJäsenet =
194            "SELECT tunnus FROM jäsenyys WHERE ryhmä='" + omatryhmat + "'";
195
196         rs1 = stmt1.executeQuery(selectJäsenet);
197
198
199         /*
200          * Käydään jokainen ryhmän jäsen läpi
201          */
202
203         while (rs1.next()) {
204
205           /*
206            * Käydään viikko läpi
207            */
```

```
208
209           for (int i = 0; i < 7; i++) {
210
211             String käsiteltäväPäivä =
212                "" + CheckDate.c(tempKalenteri.get(Calendar.DATE)) +
213                "." + CheckDate.c(tempKalenteri.get(Calendar.MONTH)+1) +
214                "." + tempKalenteri.get(Calendar.YEAR);
215
216
217             /*
218              * Haetaan käsittelyssä olevan henkilön varaukset
219              * käsiteltävälle päivälle.
220              */
221
222             String selectVaraukset =
223                "SELECT * FROM varaus WHERE tunnus ='" +
224                rs1.getString("tunnus") + "' AND pvm=to_date('" +
225                käsiteltäväPäivä + "', 'DD.MM.YYYY') " +
226                "ORDER BY alkuaika";
227
228             rs2 = stmt2.executeQuery(selectVaraukset);
229
230
231             /*
232              * Käydään yhden päivän varaukset läpi, ja vaihdetaan
233              * taulukkoon true kaikkiin varaukseen kuuluviin puolen
234              * tunnin lokeroihin.
235              */
236
237             while (rs2.next()) {
238
239               String varauksenAlku = rs2.getString("alkuaika");
240               String varauksenKesto = rs2.getString("kesto");
241               int alkuaika = 0;
242               int kesto = 0;
243
244
245               /*
246                * Muutetaan alkuaika ja varauksen kesto laskettavaan muotoon.
247                * Puolituntia on yksi rivi taulukossa. Näin olleen taulukon
248                * indeksit saadaan suoraan kertomalla tunnit kahdella ja
249                * jos ajassa on puolta tuntia merkitsevä osa, niin lisätään
250                * lukuun yksi.
251                */
252
253               if (varauksenAlku.substring(3,5).equals("30")) {
254                 alkuaika =
255                    2 * Integer.parseInt(varauksenAlku.substring(0,2)) + 1;
256               }
257               else {
258                 alkuaika =
259                    2 * Integer.parseInt(varauksenAlku.substring(0,2));
260                }
261
262                kesto =
263                   (int)((new Double(rs2.getString("kesto")).doubleValue()) * 2);
264
265
266               /*
267                * Muutetaan varauksen ajan viemät lokerot arvoksi true.
268                */
269
270               for (int j = alkuaika; j < alkuaika + kesto; j++) {
271                 ryhmänVaraukset[j][i] = true;
272               }
273
274             }  // end while rs2
275
276             tempKalenteri.add(Calendar.DATE, 1);
```

```
277
278           }  // end for
279
280           tempKalenteri.add(Calendar.DATE, -7);
281
282         }  // end while rs1
283
284
285
286       } catch (SQLException ee) {
287             out.println("Tietokantavirhe "+ee.getMessage());
288       } finally {
289             try {
290                if (rs1!=null) rs1.close();
291                if (rs2!=null) rs2.close();
292                if (stmt1!=null) stmt1.close();
293                if (stmt2!=null) stmt2.close();
294                con.close();
295             } catch(SQLException e) {
296                out.println("An SQL Exception was thrown.");
297             }
298       }
299
300
301
302       /*
303        * Tulostetaan ryhmän vapaat ajat kalenteriin
304        */
305
306       out.println("<table cellpadding='3' cellspacing='1' " +
307                   "style='background: #000000'>");
308
309
310       out.println("<tr><td class='kalenteri' colspan='8' align='center'>");
311
312       out.println("<a href='?alkupvm=" + viikkoTaakse +
313                   "&omatryhmat=" + omatryhmat + "'>&lt;&lt;</a> ");
314       out.println(" <a href='?alkupvm=" + taakse +
315                   "&omatryhmat=" + omatryhmat + "'>&lt;</a> ");
316       out.println(" <a href='?alkupvm=" + eteen +
317                   "&omatryhmat=" + omatryhmat + "'>&gt;</a> ");
318       out.println(" <a href='?alkupvm=" + viikkoEteen +
319                   "&omatryhmat=" + omatryhmat + "'>&gt;&gt;</a></td></tr>");
320
321       out.println("<tr><th>klo</th>");
322
323       for (int i = 0; i < 7; i++) {
324
325         int viikonPäivä = tempKalenteri.get(Calendar.DAY_OF_WEEK);
326
327         switch (viikonPäivä) {
328           case Calendar.MONDAY:
329             out.println("<th width='100'>Maanantai</th>"); break;
330           case Calendar.TUESDAY:
331             out.println("<th width='100'>Tiistai</th>"); break;
332           case Calendar.WEDNESDAY:
333             out.println("<th width='100'>Keskiviikko</th>"); break;
334           case Calendar.THURSDAY:
335             out.println("<th width='100'>Torstai</th>"); break;
336           case Calendar.FRIDAY:
337             out.println("<th width='100'>Perjantai</th>"); break;
338           case Calendar.SATURDAY:
339             out.println("<th width='100'>Lauantai</th>"); break;
340           case Calendar.SUNDAY:
341             out.println("<th width='100'>Sunnuntai</th>"); break;
342         }
343
344         tempKalenteri.add(Calendar.DATE, 1);
345
```

```
346       }
347
348       out.println("</tr>");
349
350       for (int i = 0; i < ryhmänVaraukset.length; i++) {
351         String klo = "";
352
353         /*
354          * Tulostetaan ajat rivien alkuun
355          */
356
357         if (i % 2 == 0) {
358           if (i < 20) klo = "0" + (i / 2) + ":00";
359           else klo = (i / 2) + ":00";
360         }
361         else {
362           klo = " ";
363         }
364
365
366         /*
367          * Varsinainen varausten ja vapaiden aikojen tulostus
368          */
369
370         out.println("<td class='kalenteri' style='font-size: 8pt; " +
371                     "padding: 0 2 0 2'>" + klo + "</td>");
372         for (int j = 0; j < ryhmänVaraukset[i].length; j++) {
373           if (ryhmänVaraukset[i][j] == true) {
374             out.println("<td height='13' style='background: #000000; " +
375                         "font-size: 4pt'> </td>");
376           }
377           else {
378             out.println("<td class='kalenteri' height='13' style='" +
379                         "font-size: 4pt'> </td>");
380           }
381         }
382         out.println("</tr>");
383       }
384
385       out.println("</table></body></html>");
386
387     }
388   }
389
390
391   private Connection createDbConnection(
392     String dbDriver, String dbServer, String dbUser, String dbPassword,
393     ServletOutputStream out) throws IOException {
394
395    // establish a database connection
396     try{
397         Class.forName(dbDriver);               // load driver
398     } catch (ClassNotFoundException e) {
399           out.println("Couldn't find driver "+dbDriver);
400           return null;
401     }
402     Connection con=null;
403     try {
404        con = DriverManager.getConnection(dbServer,dbUser,dbPassword);
405     } catch (SQLException se) {
406           out.println("Couldn\'t get connection to "+dbServer+
407                       " for "+ dbUser+"<br>");
408           out.println(se.getMessage());
409     }
410     return con;
411   }
412
413 }
```