```
 1 import java.io.*;
 2 import java.sql.*;
 3 import javax.servlet.*;
 4 import javax.servlet.http.*;
 5 import java.util.Calendar;
 6
 7 public class VuosiKalenteri extends HttpServlet {
 8
 9   final String dbDriver="org.postgresql.Driver";
10   final String dbServer ="jdbc:postgresql://localhost:10388/tsoha";
11   final String dbUser= "avtanska";        // replace with your db user account
12   final String dbPassword ="postgres"; // replace with your password
13
14
15   public void service(HttpServletRequest req, HttpServletResponse res)
16     throws ServletException, IOException {
17     ServletOutputStream out;
18     res.setContentType("text/html");
19     out= res.getOutputStream();
20
21     HttpSession session = req.getSession(false);
22     String sessionTunnus = (String)session.getValue("sessionTunnus");
23     String henkilo = req.getParameter("henkilo");
24     String ryhma = req.getParameter("ryhma");
25     String vuosi = req.getParameter("vuosi");
26
27     out.println("<html><head><title></title>" +
28                 "<link rel='stylesheet' type='text/css' href='http://" +
29                 "db.cs.helsinki.fi/u/avtanska/tsoha/vuosi.css' /><body>");
30
31
32     /*
33      * Luodaan kalenteri ja asetetaan se osoittamaan vuoden
34      * ensimmäiseen päivään.
35      */
36
37     Calendar vuosiKalenteri = Calendar.getInstance();
38
39     vuosiKalenteri.set(Calendar.YEAR, Integer.parseInt(vuosi));
40     vuosiKalenteri.set(Calendar.DAY_OF_YEAR, 1);
41
42     int päiviäRivillä = 0;
43
44     Connection con=null;
45     con= createDbConnection(dbDriver,dbServer,dbUser,dbPassword,out);
46     if (con==null) {
47        out.println("</body></html>");
48        return;
49     }
50
51     Statement stmt = null;
52     ResultSet rs = null;
53
54     out.println("<h3>" + vuosiKalenteri.get(Calendar.YEAR) + "</h3>");
55
56
57     /*
58      * Käydään kaikki 12 kuukautta läpi
59      */
60
61     for (int k = 0; k < 12; k++) {
62
63       out.println("<table cellpadding='0' cellspacing='1' " +
64                "style='background: #000000'><tr>");
65
66
67       /*
68        * Kuukauden nimi
69        */
```

```
 70
 71        switch (vuosiKalenteri.get(Calendar.MONTH)) {
 72         case Calendar.JANUARY:
 73            out.println("<td class='kuukausi' colspan='8'>Tammikuu</td>"); break;
 74         case Calendar.FEBRUARY:
 75            out.println("<td class='kuukausi'colspan='8'>Helmikuu</td>"); break;
 76         case Calendar.MARCH:
 77            out.println("<td class='kuukausi' colspan='8'>Maaliskuu</td>"); break;
 78         case Calendar.APRIL:
 79            out.println("<td class='kuukausi' colspan='8'>Huhtikuu</th>"); break;
 80         case Calendar.MAY:
 81            out.println("<td class='kuukausi' colspan='8'>Toukokuu</td>"); break;
 82         case Calendar.JUNE:
 83            out.println("<td class='kuukausi' colspan='8'>Kesäkuu</td>"); break;
 84         case Calendar.JULY:
 85            out.println("<td class='kuukausi' colspan='8'>Heinäkuu</td>"); break;
 86         case Calendar.AUGUST:
 87            out.println("<td class='kuukausi' colspan='8'>Elokuu</td>"); break;
 88         case Calendar.SEPTEMBER:
 89            out.println("<td class='kuukausi' colspan='8'>Syyskuu</td>"); break;
 90         case Calendar.OCTOBER:
 91            out.println("<td class='kuukausi' colspan='8'>Lokakuu</td>"); break;
 92         case Calendar.NOVEMBER:
 93            out.println("<td class='kuukausi' colspan='8'>Marraskuu</td>"); break;
 94         case Calendar.DECEMBER:
 95            out.println("<td class='kuukausi' colspan='8'>Joulukuu</td>"); break;
 96
 97        }
 98
 99        out.println("</tr><tr><td> </td><th>Ma</th><th>Ti</th><th>Ke</th>" +
100                    "<th>To</th><th>Pe</th><th>La</th><th>Su</th></tr><tr>");
101
102
103        /*
104         * Otetaan talteen käsittelyssä olevan kuukauden päivien
105         * lukumäärä
106         */
107
108        int maxKuunPäiviä =
109          vuosiKalenteri.getActualMaximum(Calendar.DAY_OF_MONTH);
110        int kuunPäivä = 1;
111        String varattu = "";
112        String linkki = "";
113        String pvm = "";
114        String päivä = "";
115
116
117        /*
118         * Käydään kuukauden päivät läpi.
119         */
120
121        while (kuunPäivä <= maxKuunPäiviä) {
122          if (päiviäRivillä == 0) {
123            out.println("<th class='viikko'>" +
124                        vuosiKalenteri.get(Calendar.WEEK_OF_YEAR) + "</th>");
125          }
126
127
128          /*
129           * Jos kuukauden ensimmäinen viikko ei ala maanantaista, tulostetaan
130           * tyhjiä soluja sen viikon riville
131           */
132
133          if (vuosiKalenteri.get(Calendar.DAY_OF_MONTH) == 1) {
134            Calendar temp = (Calendar)vuosiKalenteri.clone();
135            while (temp.get(Calendar.DAY_OF_WEEK) != Calendar.MONDAY) {
136              out.println("<td> </td>");
137              temp.add(Calendar.DAY_OF_WEEK, -1);
138              päiviäRivillä++;
```

```
139           }
140         }
141
142
143         /*
144          * Päivä-muuttujan arvo tulostetaan kalenteriin jokaisen päivän
145          * kohdalle. Jos päivälle on varauksia, niin lisätään päivän
146          * numeron yhteyteen linkki viikkokalenteriin.
147          */
148
149         päivä = "" + vuosiKalenteri.get(Calendar.DAY_OF_MONTH);
150
151
152         pvm = CheckDate.c(vuosiKalenteri.get(Calendar.DAY_OF_MONTH)) + "." +
153               CheckDate.c((vuosiKalenteri.get(Calendar.MONTH)+1)) + "." +
154               vuosiKalenteri.get(Calendar.YEAR);
155
156         stmt = null;
157         rs = null;
158
159
160         /*
161          * Tarkistetaan jokaisen päivän kohdalla löytyykö henkilön
162          * kalenterista yhtäkään varausta sille päivälle
163          */
164
165         try {
166           stmt = con.createStatement();
167           if (!henkilo.equals("null")) {
168             rs = stmt.executeQuery(
169               "SELECT 1 as true FROM varaus WHERE tunnus='" + henkilo +
170               "' AND pvm=to_date('" + pvm + "', 'DD.MM.YYYY')");
171           }
172           else if (!ryhma.equals("null")) {
173             rs = stmt.executeQuery(
174               "SELECT 1 as true FROM varaus WHERE ryhmä='" + ryhma +
175               "' AND pvm=to_date('" + pvm + "', 'DD.MM.YYYY')");
176           }
177
178           if(rs.next()) {
179             varattu = "class='varaus'";
180             if (!henkilo.equals("null")) {
181               päivä = "<a href='http://db.cs.helsinki.fi/s/avtanska/" +
182                       "Kalenteri?alkupvm=" + pvm + "&henkilo=" + henkilo +
183                       "&viikkoAlusta=true'>" + päivä + "</a>";
184             }
185             else if (!ryhma.equals("null")) {
186               päivä = "<a href='http://db.cs.helsinki.fi/s/avtanska/" +
187                       "Kalenteri?alkupvm=" + pvm + "&ryhma=" + ryhma +
188                       "&viikkoAlusta=true'>" + päivä + "</a>";
189             }
190           }
191         } catch (SQLException ee) {
192           out.println("Tietokantavirhe "+ee.getMessage());
193         }
194
195
196         out.println("<td " + varattu + ">" + päivä  + "</td>");
197
198         varattu = "";
199
200         päiviäRivillä++;
201
202         /*
203          * Toiminta viikon viimeisen päivän kohdalla. Jos ollaan
204          * sunnuntaissa, joka on myös kuukauden viimeinen päivä,
205          * ei enää aloiteta uutta riviä taulukossa.
206          *
207          * Jos päivä on sunnuntai, mutta kuukausi ei ole lopussa,
```

```
208          * aloitetaan uusi rivi seuraavaa viikkoa varten.
209          */
210
211         if (vuosiKalenteri.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY &&
212             vuosiKalenteri.get(Calendar.DAY_OF_MONTH) ==
213             vuosiKalenteri.getActualMaximum(Calendar.DAY_OF_MONTH)) {
214           out.println("</tr>");
215           päiviäRivillä = 0;
216         }
217         else if (vuosiKalenteri.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY) {
218           out.println("</tr><tr>");
219           päiviäRivillä = 0;
220         }
221
222
223         /*
224          * Jos kuukauden viimeinen päivä ei ole sunnuntai, täytetään
225          * loppuviikko tyhjillä soluilla
226          */
227
228         if (vuosiKalenteri.get(Calendar.DAY_OF_MONTH) ==
229             vuosiKalenteri.getActualMaximum(Calendar.DAY_OF_MONTH) &&
230             vuosiKalenteri.get(Calendar.DAY_OF_WEEK) != Calendar.SUNDAY) {
231
232           Calendar temp2 = (Calendar)vuosiKalenteri.clone();
233           while (päiviäRivillä < 7) {
234             out.println("<td> </td>");
235             temp2.add(Calendar.DAY_OF_WEEK, 1);
236             päiviäRivillä++;
237           }
238           päiviäRivillä = 0;
239           out.println("</tr>");
240         }
241
242
243         vuosiKalenteri.add(Calendar.DAY_OF_MONTH, 1);
244         kuunPäivä++;
245       } // end while
246
247       out.println("</table><br /><br />");
248
249     } // end for k
250
251
252     /*
253      * Suljetaan tietokantayhteys
254      */
255
256     try {
257       if (rs!=null) rs.close();
258       if (stmt!=null) stmt.close();
259       con.close();
260     } catch(SQLException e) {
261        out.println("An SQL Exception was thrown.");
262     }
263
264   }
265
266
267   private Connection createDbConnection(
268     String dbDriver, String dbServer, String dbUser, String dbPassword,
269     ServletOutputStream out) throws IOException {
270
271     // establish a database connection
272     try{
273         Class.forName(dbDriver);               // load driver
274     } catch (ClassNotFoundException e) {
275           out.println("Couldn't find driver "+dbDriver);
276           return null;
```

```
277     }
278     Connection con=null;
279     try {
280        con = DriverManager.getConnection(dbServer,dbUser,dbPassword);
281     } catch (SQLException se) {
282           out.println("Couldn\'t get connection to "+dbServer+
283                       " for "+ dbUser+"<br>");
284           out.println(se.getMessage());
285     }
286     return con;
287   }
288
289 }
```