```
 2 * MainDialog.java
 6
 7 package info.stastny.cpbr;
 8
 9 import java.awt.Desktop;
37
38 /**
39  *
40  * @author Marek Stastny, http://stastny.info
41  */
42 public class MainDialog extends javax.swing.JDialog {
43     private static final long serialVersionUID = 1L;
44     private static String dbPath="", csvPath="";
45     private static DataProtector dp= new DataProtector();
46     private static String hpURL="http://dstats.net/fwd/ub1bz";
47
48     /** Creates new form MainDialog */
49     public MainDialog(java.awt.Frame parent, ModalityType toolkitModal) {
50         super(parent, toolkitModal);
51         initComponents();
52     }
53
54     /** This method is called from within the constructor to
55      * initialize the form.
56      * WARNING: Do NOT modify this code. The content of this method is
57      * always regenerated by the Form Editor.
58      */
61     private void initComponents() {
192
193     private void btDBActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_btDBActionPerformed
194         JFileChooser fc = new JFileChooser(dbPath);
195         fc.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
196         if (fc.showOpenDialog(this)==JFileChooser.APPROVE_OPTION){
197             try {
198                 dbPath=fc.getSelectedFile().getCanonicalPath();
199                 tfDB.setText(dbPath);
200             } catch (IOException e) {
201                 errorPanel("Failed to locate profile directory!");
202             }
203         }
204     }//GEN-LAST:event_btDBActionPerformed
205
206     private void btCSVActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_btCSVActionPerformed
207         JFileChooser fc = new JFileChooser(csvPath);
208         fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
209         if (fc.showOpenDialog(this)==JFileChooser.APPROVE_OPTION){
210             try {
211                 csvPath=fc.getSelectedFile().getCanonicalPath();
212                 tfCSV.setText(csvPath);
213             } catch (IOException e) {
214                 errorPanel("Failed to locate CSV file!");
215             }
216         }
217     }//GEN-LAST:event_btCSVActionPerformed
218
219     private void btExportActionPerformed(java.awt.event.ActionEvent evt)
    {//GEN-FIRST:event_btExportActionPerformed
220         try {
```

```
221             CSVWriter writer=new CSVWriter(new FileWriter(tfCSV.getText()));
222             Connection
    conn=DriverManager.getConnection("jdbc:sqlite:"+dbPath+File.separatorChar+"Login
    Data");
223
224             Statement st=conn.createStatement();
225             ResultSet rs=st.executeQuery("select * from logins;");
226             ResultSetMetaData rsmd=rs.getMetaData();
227             int columns=rsmd.getColumnCount();
228             int rows=0;
229             int passindex=-1;
230             String[] rsa=new String[columns];
231
232             //1.line = column names
233             for(int i=1;i<=columns;i++){
234                 rsa[i-1]=rsmd.getColumnName(i);
235                 if (rsmd.getColumnName(i).equals("password_value"))
236                     passindex=i;
237             }
238             writer.writeNext(rsa);
239
240             //Export to CSV
241             while (rs.next()) {
242                 for(int i=1;i<=columns;i++){
243                     if (i!=passindex)
244                         rsa[i-1]=rs.getString(i);
245                     else{
246                         rsa[i-1]=dp.unprotect(rs.getBytes(passindex));
247                     }
248                 }
249                 writer.writeNext(rsa);
250                 rows++;
251             }
252             writer.close();
253             conn.close();
254
255             infoPanel("Backup successfully completed ("+rows+" record"+
    (rows==1?"":"s")+")!");
256         } catch (Exception e) {
257             errorPanel(e.getMessage()!=null?e.getMessage():e.toString());
258         }//finally...
259
260     }//GEN-LAST:event_btExportActionPerformed
261
262     private void btImportActionPerformed(java.awt.event.ActionEvent evt)
     {//GEN-FIRST:event_btImportActionPerformed
263         if (JOptionPane.showConfirmDialog(this, "Do You really want to restore data
    from following file?\n"+csvPath, null, JOptionPane.YES_NO_OPTION,
     JOptionPane.QUESTION_MESSAGE)==JOptionPane.YES_OPTION){
264             try{
265                 String db=dbPath+File.separatorChar+"Login Data";
266                 CSVReader reader=new CSVReader(new FileReader(tfCSV.getText()));
267                 Connection conn=DriverManager.getConnection("jdbc:sqlite:"+db);
268                 Statement st;
269                 PreparedStatement ps;
270                 String[] rsa, rsh;
271                 int i=0, passindex=-1, rows=0;
272
273                 //Backup Login Data file
274                 copyFile(db, db+String.format(" %ts", new Date()));
```

```
275
276                 //Read column names
277                 rsh=reader.readNext();
278                 i=0;
279                 for(String s:rsh){
280                     if (s.equals("password_value")){
281                         passindex=i;
282                         break;
283                     }
284                     i++;
285                 }
286
287                 //Process CSV
288                 ps = conn.prepareStatement("insert into logins values("+new
    String(new char[rsh.length-1]).replace("\0","?,")+"?)");
289                 conn.setAutoCommit(false);
290                 st = conn.createStatement();
291                 st.executeUpdate("delete from logins");
292
293                 while((rsa=reader.readNext())!=null){
294                     i=0;
295                     for(String s:rsa){
296                         if (i++==passindex){
297                             ps.setBytes(i, dp.protect(s));
298                         }else{
299                             ps.setString(i, s);
300                         }
301                     }
302                     ps.executeUpdate();
303                     rows++;
304                 }
305                 conn.commit();
306
307                 reader.close();
308                 conn.close();
309
310                 infoPanel("Restore successfully completed ("+rows+" record"+
    (rows==1?"":"s")+")!");
311             }catch(Exception e){
312                 errorPanel(e.getMessage()!=null?e.getMessage():e.toString());
313             }//finally...
314
315         }
316     }//GEN-LAST:event_btImportActionPerformed
317
318     private void onNoticeClicked(java.awt.event.MouseEvent evt)
     {//GEN-FIRST:event_onNoticeClicked
319         String errmsg="Cannot open homepage, please start browser and enter
    following URL into address bar:\n\n"+hpURL;
320         if (Desktop.isDesktopSupported()) {
321             Desktop desktop = Desktop.getDesktop();
322             try {
323                 desktop.browse(URI.create((hpURL)));
324             } catch (IOException e) {
325                 errorPanel(errmsg);
326             }
327         } else {
328             errorPanel(errmsg);
329         }
330     }//GEN-LAST:event_onNoticeClicked
```

```
331
332     void copyFile(String source, String dest) throws IOException {
333         int size=-1;
334         byte[] buffer=new byte[1024];
335
336         File destFile=new File(dest);
337         if (!destFile.exists())
338             destFile.createNewFile();
339
340         InputStream in=new FileInputStream(new File(source));
341         OutputStream out=new FileOutputStream(destFile);
342
343         while ((size=in.read(buffer))>0) {
344             out.write(buffer, 0, size);
345         }
346
347         in.close();
348         out.close();
349     }
350
351     void errorPanel(String message){
352         JOptionPane.showMessageDialog(this, message, null,
    JOptionPane.ERROR_MESSAGE);
353     }
354
355     void infoPanel(String message){
356         JOptionPane.showMessageDialog(this, message, null,
    JOptionPane.INFORMATION_MESSAGE);
357     }
358
359     /**
360     * @param args the command line arguments
361     */
362     public static void main(String args[]) {
363         try{
364             System.load(new File (".").getCanonicalPath()
   +File.separatorChar+"jdpapi-native.dll");
365             Class.forName("org.sqlite.JDBC");
366         }catch(Exception e){
367             JOptionPane.showMessageDialog(null, "Initialization failed!", "Fatal",
    JOptionPane.ERROR_MESSAGE);
368             System.exit(-1);
369         }
370
371         java.awt.EventQueue.invokeLater(new Runnable() {
372             public void run() {
373                 try{
374                     JFrame.setDefaultLookAndFeelDecorated(true);
375                     MainDialog dialog = new MainDialog((JFrame)null,
    ModalityType.TOOLKIT_MODAL);
376                     dialog.tpInfo.setText(" 1. You have to close Chrome browser
   before doing anything!\n 2. Backup file will contain decrypted passwords in plain
   text, keep it safe!\n 3. Backup Login Data file located in Chrome profile directory
   before restoring!\n 4. This application is freeware, provided 'as is' without
   warranty of any kind!\n 5. Use at Your own Risk! If You are unsure, don't use this
   application!\n\n Thirdparty Software:\n JDPAPI.sf.net, zentus.com/SQLiteJDBC,
   OpenCSV.sf.net, Launch4J.sf.net");
377                     dialog.lbNotice.setText("(c)2011 Marek Stastny. Click here to
   visit homepage of Chrome Passwords Backup&Restore.");
378
```

```
   UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
379                     SwingUtilities.updateComponentTreeUI(dialog);
380                     dialog.addWindowListener(new java.awt.event.WindowAdapter() {
381                         public void windowClosing(java.awt.event.WindowEvent e) {
382                             System.exit(0);
383                         }
384                     });
385 
386                     Dimension paneSize = dialog.getSize();
387                     Dimension screenSize = dialog.getToolkit().getScreenSize();
388                     dialog.setLocation((screenSize.width-paneSize.width)/2, (int)
   ((screenSize.height-paneSize.height)*0.45));
389 
390                     //INIT
391                     String[] profileDir=new String[3];
392                     profileDir[0]=System.getenv("USERPROFILE")+"\\Local
   Settings\\Application Data\\Google\\Chrome\\User Data\\Default";
393                     profileDir[1]=System.getenv("USERPROFILE")+"\\Local
   Settings\\Data aplikací\\Google\\Chrome\\User Data\\Default";
394                     profileDir[2]=System.getenv("USERPROFILE")
   +"\\AppData\\Local\\Google\\Chrome\\User Data\\Default";
395                     for(String pd:profileDir){
396                         if (new File(pd).exists()){
397                             dbPath=pd;
398                             break;
399                         }
400                     }
401                     if (dbPath.length()==0){
402                         dbPath=System.getenv("USERPROFILE");
403                         dialog.errorPanel("Chrome Profile directory not
   auto-detected! You will need to locate it manually.");
404                     }
405                     dialog.tfDB.setText(dbPath);
406 
407                     csvPath=new
    JFileChooser().getFileSystemView().getDefaultDirectory().getCanonicalPath()
   +File.separatorChar+"Login Data.csv";
408                     dialog.tfCSV.setText(csvPath);
409 
410                     dialog.setVisible(true);
411                 } catch (Exception e) {
412                     JOptionPane.showMessageDialog(null, "Initialization failed!",
    "Fatal", JOptionPane.ERROR_MESSAGE);
413                 }
414             }
415         });
416     }
417 
418     // Variables declaration - do not modify//GEN-BEGIN:variables
419     private javax.swing.JButton btCSV;
420     private javax.swing.JButton btDB;
421     private javax.swing.JButton btExport;
422     private javax.swing.JButton btImport;
423     private javax.swing.JLabel jLabel1;
424     private javax.swing.JLabel jLabel2;
425     private javax.swing.JLabel jLabel3;
426     private javax.swing.JScrollPane jScrollPane1;
427     private javax.swing.JLabel lbNotice;
428     private javax.swing.JTextField tfCSV;
429     private javax.swing.JTextField tfDB;
```

```
430     private javax.swing.JTextPane tpInfo;
431     // End of variables declaration//GEN-END:variables
432 }
433
```