

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIROAdaptorConfigurationList.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.configuration;
31:
32: import eu.biro.adaptor2.BIROAdaptorConfiguration;
33: import eu.biro.adaptor2.DBMSDriver;
34: import java.io.File;
35: import java.io.FileInputStream;
36: import java.io.FileOutputStream;
37: import java.io.ObjectInputStream;
38: import java.io.ObjectOutputStream;
39: import java.util.HashMap;
40: import java.util.Iterator;
41: import java.util.LinkedList;
42: import java.util.List;
43: import java.util.Map;
44:
45: public class BIROAdaptorConfigurationList implements java.io.Serializable {
```

```
46:
47:     private static final long serialVersionUID = 201812182229L;
48:     private static BIROAdaptorConfigurationList bIROAdaptorConfigurationList;
49:     private List<BIROAdaptorConfiguration> configurationList;
50:     private final Map<String, BIROAdaptorConfiguration> nameConfigurationMap;
51:     private BIROAdaptorConfiguration currentConfiguration;
52:
53:     private BIROAdaptorConfigurationList() {
54:         configurationList = new LinkedList<BIROAdaptorConfiguration>();
55:         nameConfigurationMap = new HashMap<String, BIROAdaptorConfiguration>();
56:
57:     }
58:
59:     public void addBIROAdaptorConfiguration(BIROAdaptorConfiguration configuration) {
60:         configurationList.add(configuration);
61:         nameConfigurationMap.put(configuration.getName(), configuration);
62:
63:     }
64:
65:     public boolean containsConfigurationName(String name) {
66:         return nameConfigurationMap.containsKey(name);
67:     }
68:
69:     public BIROAdaptorConfiguration getCurrentConfiguration() {
70:         return currentConfiguration;
71:     }
72:
73:     public void setCurrentConfiguration(BIROAdaptorConfiguration currentConfiguration) {
74:         this.currentConfiguration = currentConfiguration;
75:     }
76:
77:     public List<BIROAdaptorConfiguration> getConfigurationList() {
78:         return configurationList;
79:     }
80:
81:     public BIROAdaptorConfiguration[] getConfigurationArray() {
82:         return configurationList.toArray(new BIROAdaptorConfiguration[configurationList.size()]);
83:     }
84:
85:     private void clear() {
86:         configurationList.clear();
87:         nameConfigurationMap.clear();
88:
89:     }
90:
```

```
91: public void removeConfiguration(BIROAdaptorConfiguration conf) {
92:     nameConfigurationMap.remove(conf.getName());
93:     configurationList.remove(conf);
94: }
95:
96: public String[] getConfigurationNames() {
97:     String[] s = (nameConfigurationMap.keySet()).toArray(new String[nameConfigurationMap.size()]);
98:     return s;
99: }
100:
101: public BIROAdaptorConfiguration getFieldWithName(String name) {
102:     return nameConfigurationMap.get(name);
103: }
104:
105: public int getDBMSDriverUsersNumber(DBMSDriver dbMSDriver) {
106:     int users = 0;
107:     DBMSDriver d;
108:     Iterator<BIROAdaptorConfiguration> i = configurationList.iterator();
109:     while (i.hasNext()) {
110:         d = i.next().getDBMSDriver();
111:         if (d.compare(dbMSDriver)) {
112:             users++;
113:         }
114:     }
115:     return users;
116: }
117:
118: public void saveToFile(File f) throws Exception {
119:     FileOutputStream fos;
120:     ObjectOutputStream oos = null;
121:     try {
122:         fos = new FileOutputStream(f);
123:         oos = new ObjectOutputStream(fos);
124:         oos.writeObject(this);
125:         oos.flush();
126:     } catch (Exception e) {
127:         throw e;
128:     } finally {
129:         if (oos != null) {
130:             oos.close();
131:         }
132:     }
133: }
134:
135:
```

BIROBox/src/eu/biro/birobox/configuration/BIROAdaptorConfigurationList.java

```
136: public static BIROAdaptorConfigurationList readFromFile(File f) throws Exception {
137:     FileInputStream fis;
138:     ObjectInputStream ois = null;
139:     try {
140:         fis = new FileInputStream(f);
141:         ois = new ObjectInputStream(fis);
142:         BIROAdaptorConfigurationList conf = (BIROAdaptorConfigurationList) ois.readObject();
143:         return conf;
144:     } catch (Exception e) {
145:         throw e;
146:     } finally {
147:         if (ois != null) {
148:             ois.close();
149:         }
150:     }
151: }
152:
153: public static BIROAdaptorConfigurationList getBIROAdaptorConfigurationList() {
154:     if (bIROAdaptorConfigurationList == null) {
155:         throw new IllegalStateException("Devi caricare prima la configurazione");
156:     }
157:     return bIROAdaptorConfigurationList;
158: }
159:
160: public static void newBIROAdaptorConfigurationList(File f) throws Exception {
161:     if (f.exists()) {
162:         bIROAdaptorConfigurationList = readFromFile(f);
163:     } else {
164:         bIROAdaptorConfigurationList = new BIROAdaptorConfigurationList();
165:     }
166: }
167: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIROBoxConfiguration.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.configuration;
31:
32: import eu.biro.adaptor2.DBMSDriver;
33: import eu.biro.adaptor2.Drivers.JDBCDBCdriver;
34: import eu.biro.adaptor2.Drivers.MSSQLSERVER20002005Driver;
35: import eu.biro.adaptor2.Drivers.MSSQLSERVER2008Driver;
36: import eu.biro.adaptor2.Drivers.MYSQLDriver;
37: import eu.biro.adaptor2.Drivers.ORACLE10Driver;
38: import eu.biro.adaptor2.Drivers.POSTGRESQlDriver;
39: import eu.biro.adaptor2.Drivers.SYBASEDriver;
40:
41: import java.io.File;
42: import java.io.FileInputStream;
43: import java.io.FileOutputStream;
44: import java.io.ObjectOutputStream;
45: import java.io.ObjectInputStream;
```

```
46: import java.io.Serializable;
47:
48: import java.util.Vector;
49:
50:
51: public class BIROBoxConfiguration implements Serializable {
52:
53:     private static final long serialVersionUID = -4872666212438412512L;
54:     private Vector<DBMSDriver> dbMSDriverVector;
55:     private static BIROBoxConfiguration dbMSVectorConfiguration;
56:
57:     private BIROBoxConfiguration() {
58:         dbMSDriverVector = new Vector<DBMSDriver>();
59:         dbMSDriverVector.add(new POSTGRESQLDriver());
60:         dbMSDriverVector.add(new MySQLDriver());
61:         dbMSDriverVector.add(new MSSQLSERVER20002005Driver());
62:         dbMSDriverVector.add(new MSSQLSERVER2008Driver());
63:         dbMSDriverVector.add(new ORACLE10Driver());
64:         dbMSDriverVector.add(new SYBASEDriver());
65:     }
66:
67:     public Vector<DBMSDriver> getDBMSDriverVector() {
68:         return dbMSDriverVector;
69:     }
70:
71:     public void setDBMSDriverVector(Vector<DBMSDriver> dbMSDriverVector) {
72:         this.dbMSDriverVector = dbMSDriverVector;
73:     }
74:
75:
76:     public void addDBMSDriver(DBMSDriver dbMSDriver) {
77:         dbMSDriverVector.add(dbMSDriver);
78:     }
79:
80:     public void saveToFile(File file) throws Exception {
81:         FileOutputStream fos;
82:         ObjectOutputStream oos = null;
83:         try {
84:             fos = new FileOutputStream(file);
85:             oos = new ObjectOutputStream(fos);
86:             oos.writeObject(this);
87:             oos.flush();
88:         } catch (Exception e) {
89:             throw e;
90:         } finally {
```

```
91:         if (oos != null) {
92:             oos.close();
93:         }
94:     }
95: }
96:
97: private static BIROBoxConfiguration readFromFile(File file) throws Exception {
98:     FileInputStream fis;
99:     ObjectInputStream ois = null;
100:     try {
101:         fis = new FileInputStream(file);
102:         ois = new ObjectInputStream(fis);
103:         BIROBoxConfiguration conf = (BIROBoxConfiguration) ois.readObject();
104:         return conf;
105:     } catch (Exception e) {
106:         throw e;
107:     } finally {
108:         if (ois != null) {
109:             ois.close();
110:         }
111:     }
112: }
113:
114: public static BIROBoxConfiguration getBIROBoxConfiguration() {
115:     if (dbMSVectorConfiguration == null) {
116:         throw new IllegalStateException("Devi caricare prima la configurazione");
117:     }
118:     return dbMSVectorConfiguration;
119: }
120:
121: public static void newBIROBoxConfiguration(File f) throws Exception {
122:     if (f.exists()) {
123:         dbMSVectorConfiguration = readFromFile(f);
124:     } else {
125:         dbMSVectorConfiguration = new BIROBoxConfiguration();
126:     }
127: }
128: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIRODatabaseManagerConfiguration.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.configuration;
31:
32: import eu.biro.adaptor2.DBMSDriver;
33: import eu.biro.adaptor2.field.BIROFieldList;
34: import java.io.File;
35: import java.io.FileInputStream;
36: import java.io.FileOutputStream;
37: import java.io.ObjectInputStream;
38: import java.io.ObjectOutputStream;
39: import java.io.Serializable;
40:
41: public class BIRODatabaseManagerConfiguration implements Serializable {
42:
43:     private static BIRODatabaseManagerConfiguration biRODatabaseManagerConfiguration;
44:     private DBMSDriver biRODBMSDriver;
45:     private String databaseManagerInputDirectory;
```

```
46: private BIROFieldList biROFieldList;
47: private static final long serialVersionUID = -7165362834519202703L;
48:
49: private BIRODatabaseManagerConfiguration() {
50:     this.bIRODBMSDriver = null;
51:     this.databaseManagerInputDirectory = "";
52:     this.bIROFieldList = BIROFieldList.newBIROFieldList();
53: }
54:
55:
56: public void setDatabaseManagerInputDirectory(String databaseManagerInputDirectory) {
57:     this.databaseManagerInputDirectory = databaseManagerInputDirectory;
58: }
59:
60: public void setBIROFieldList(BIROFieldList biroFieldList) {
61:     this.bIROFieldList = biroFieldList;
62: }
63:
64:
65:
66: public void setBIRODBMSDriver(DBMSDriver bIRODBMSDriver) {
67:     this.bIRODBMSDriver = bIRODBMSDriver;
68: }
69:
70: public BIROFieldList getBIROFieldList() {
71:     return biROFieldList;
72: }
73:
74:
75: public String getDatabaseManagerInputDirectory() {
76:     return databaseManagerInputDirectory;
77: }
78:
79: public DBMSDriver getBIRODBMSDriver() {
80:     return bIRODBMSDriver;
81: }
82:
83: public void saveToFile(File file) throws Exception {
84:     FileOutputStream fos;
85:     ObjectOutputStream oos = null;
86:     try {
87:         fos = new FileOutputStream(file);
88:         oos = new ObjectOutputStream(fos);
89:         oos.writeObject(this);
90:         oos.flush();
```

```
91:         } catch (Exception e) {
92:             throw e;
93:         } finally {
94:             if (oos != null) {
95:                 oos.close();
96:             }
97:         }
98:     }
99:
100: private static BIRODatabaseManagerConfiguration readFromFile(File file) throws Exception {
101:     FileInputStream fis;
102:     ObjectInputStream ois = null;
103:     try {
104:         fis = new FileInputStream(file);
105:         ois = new ObjectInputStream(fis);
106:         BIRODatabaseManagerConfiguration conf = (BIRODatabaseManagerConfiguration) ois.readObject();
107:         return conf;
108:     } catch (Exception e) {
109:         throw e;
110:     } finally {
111:         if (ois != null) {
112:             ois.close();
113:         }
114:     }
115: }
116:
117: public static BIRODatabaseManagerConfiguration getBIRODatabaseManagerConfiguration() {
118:     if (bIRODatabaseManagerConfiguration == null) {
119:         throw new IllegalStateException("Devi caricare prima la configurazione");
120:     }
121:     return bIRODatabaseManagerConfiguration;
122: }
123:
124: public static void newBIRODatabaseManagerConfiguration(File f) throws Exception {
125:     if (f.exists()) {
126:         try {
127:             bIRODatabaseManagerConfiguration = readFromFile(f);
128:         } catch (Exception e) {
129:             bIRODatabaseManagerConfiguration = new BIRODatabaseManagerConfiguration();
130:         }
131:     } else {
132:         bIRODatabaseManagerConfiguration = new BIRODatabaseManagerConfiguration();
133:     }
134: }
135: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      CommunicationSoftwareConfiguration.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.configuration;
31:
32: import eu.biro.birobox.panel.communicationSoftware.StatisticalObject;
33: import java.io.File;
34: import java.io.FileInputStream;
35: import java.io.FileOutputStream;
36: import java.io.ObjectOutputStream;
37: import java.io.ObjectInputStream;
38: import java.io.Serializable;
39: import java.text.ParseException;
40: import java.util.Enumeration;
41: import java.util.Vector;
42:
43: public class CommunicationSoftwareConfiguration implements Serializable {
44:
45:     private static final long serialVersionUID = -4873666220438411519L;
```

```
46: private Vector<StatisticalObject> statisticalObjects;
47: private static CommunicationSoftwareConfiguration communicationSoftwareConfiguration;
48: private String centralBIROSystemIPAddress;
49: private String webServiceEndpoint;
50:
51: private CommunicationSoftwareConfiguration() throws ParseException {
52:     statisticalObjects = new Vector<StatisticalObject>();
53:     this.centralBIROSystemIPAddress = "";
54:     this.webServiceEndpoint="";
55:
56: }
57:
58: public Vector<StatisticalObject> getStatisticalObjects() {
59:     return statisticalObjects;
60: }
61:
62: public void setStatisticalObjects(Vector<StatisticalObject> statisticalObjects) {
63:     this.statisticalObjects = statisticalObjects;
64: }
65:
66: public String getCentralBIROSystemIPAddress() {
67:     return this.centralBIROSystemIPAddress;
68: }
69:
70: public void setCentralBIROSystemIPAddress(String centralBIROSystemIPAddress) {
71:     this.centralBIROSystemIPAddress = centralBIROSystemIPAddress;
72: }
73:
74: public String getWebServiceEndpoint() {
75:     return webServiceEndpoint;
76: }
77:
78: public void setWebServiceEndpoint(String webServiceEndpoint) {
79:     this.webServiceEndpoint = webServiceEndpoint;
80: }
81:
82: public boolean containsStatisticalObject(String statisticalObjectID) {
83:     boolean found = false;
84:     Enumeration<StatisticalObject> e = statisticalObjects.elements();
85:     while (e.hasMoreElements() && !found) {
86:         if (e.nextElement().getId().equals(statisticalObjectID)) {
87:             found = true;
88:         }
89:     }
90:     return found;
```

```
91:     }
92:
93:     public void saveToFile(File file) throws Exception {
94:         FileOutputStream fos;
95:         ObjectOutputStream oos = null;
96:         try {
97:             fos = new FileOutputStream(file);
98:             oos = new ObjectOutputStream(fos);
99:             oos.writeObject(this);
100:            oos.flush();
101:        } catch (Exception e) {
102:            throw e;
103:        } finally {
104:            if (oos != null) {
105:                oos.close();
106:            }
107:        }
108:    }
109:
110:    private static CommunicationSoftwareConfiguration readFromFile(File file) throws Exception {
111:        FileInputStream fis;
112:        ObjectInputStream ois = null;
113:        try {
114:            fis = new FileInputStream(file);
115:            ois = new ObjectInputStream(fis);
116:            CommunicationSoftwareConfiguration conf = (CommunicationSoftwareConfiguration) ois.readObject();
117:            return conf;
118:        } catch (Exception e) {
119:            throw e;
120:        } finally {
121:            if (ois != null) {
122:                ois.close();
123:            }
124:        }
125:    }
126:
127:    public static CommunicationSoftwareConfiguration getCommunicationSoftwareConfiguration() {
128:        if (communicationSoftwareConfiguration == null) {
129:            throw new IllegalStateException("Devi caricare prima la configurazione");
130:        }
131:        return communicationSoftwareConfiguration;
132:    }
133:
134:    public static void newCommunicationSoftwareConfiguration(File f) throws Exception {
135:        if (f.exists()) {
```

```
136:         try {
137:             communicationSoftwareConfiguration = readFromFile(f);
138:         } catch (Exception e) {
139:             communicationSoftwareConfiguration = new CommunicationSoftwareConfiguration();
140:         }
141:     } else {
142:         communicationSoftwareConfiguration = new CommunicationSoftwareConfiguration();
143:     }
144: }
145: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      Configuration.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10: * the Free Software Foundation; either version 2, or (at your option)
11: * any later version.
12: *
13: * This file is distributed in the hope that it will be useful,
14: * but WITHOUT ANY WARRANTY; without even the implied warranty of
15: * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16: * GNU General Public License for more details.
17: *
18: * You should have received a copy of the GNU General Public License
19: * along with this file; see the file COPYING. If not, write to
20: * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21: *
22: * In short: you may use this file any way you like, as long as you
23: * don't charge money for it, remove this notice, or hold anyone liable
24: * for its results.
25: *
26:
27: * GPL Copyright, The BIRO Project
28: *
29: **/
30: package eu.biro.birobox.configuration;
31:
32: import java.io.File;
33: import java.net.URI;
34:
35:
36: public class Configuration {
37:
38:     public static final String BIROAdaptorConfigurationListFilePath =
39: "BIROBox/conf/BIROAdaptorConfigurationListFile.dat";
40:     public static final String BIRODatabaseManagerConfigurationFilePath =
41: "BIROBox/conf/BIRODatabaseManagerConfigurationFile.dat";
42:     public static final String BIROBoxConfigurationFilePath = "BIROBox/conf/BIROBoxConfigurationFile.dat";
43:     public static final String statisticalEngineConfigurationPath =
44: "BIROBox/conf/statisticalEngineConfigurationFile.dat";
45:     public static final String CommunicationSoftwareConfigurationPath =
```

```
"BIROBox/conf/BIROCommunicationSoftwareConfigurationFile.dat";
43:     public static final String root = (new File("").getAbsolutePath()).replaceAll("\\\\", "/").concat("/");
44:     public static final String bIROAdaptorDefaultOutputFolder = "_de_/output/xml/";
45: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEngineConfiguration.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.configuration;
31:
32: import java.io.File;
33: import java.io.FileInputStream;
34: import java.io.FileOutputStream;
35: import java.io.ObjectOutputStream;
36: import java.io.ObjectInputStream;
37: import java.io.Serializable;
38: import java.util.Calendar;
39: import java.util.Date;
40:
41:
42: public class StatisticalEngineConfiguration implements Serializable {
43:
44:     private static final long serialVersionUID = -4873666220438411519L;
45:     private int centreID;
```

```
46:     private int yearNow;
47:     private int startYear;
48:     private int duration;
49:     private String refAnaDate;
50:     private String populationFilePath;
51:     private String diabeticPopulationFilePath;
52:     private static StatisticalEngineConfiguration statisticalEngineConfiguration;
53:
54:     private StatisticalEngineConfiguration() {
55:         int currentYear = (Calendar.getInstance()).get(Calendar.YEAR);
56:         this.centreID = 1;
57:         this.yearNow = currentYear;
58:         this.startYear = currentYear;
59:         this.duration = 1;
60:         this.populationFilePath = "";
61:         this.diabeticPopulationFilePath = "";
62:
63:     }
64:
65:     public int getCentreID() {
66:         return centreID;
67:     }
68:
69:     public int getStartYear() {
70:         return startYear;
71:     }
72:
73:     public int getDuration() {
74:         return duration;
75:     }
76:
77:     public int getYearNow() {
78:         return yearNow;
79:     }
80:
81:     public String getDiabeticPopulationFilePath() {
82:         return diabeticPopulationFilePath;
83:     }
84:
85:     public String getPopulationFilePath() {
86:         return populationFilePath;
87:     }
88:
89:     public String getRefAnaDate() {
90:         return refAnaDate;
```

```
91:     }
92:
93:     public void setCentreID(int centreID) {
94:         this.centreID = centreID;
95:     }
96:
97:     public void setStartYear(int startYear) {
98:         this.startYear = startYear;
99:     }
100:
101:     public void setDuration(int duration) {
102:         this.duration = duration;
103:     }
104:
105:
106:     public void setYearNow(int yearNow) {
107:         this.yearNow = yearNow;
108:     }
109:
110:     public void setDiabeticPopulationFilePath(String diabeticPopulationFilePath) {
111:         this.diabeticPopulationFilePath = diabeticPopulationFilePath;
112:     }
113:
114:     public void setPopulationFilePath(String populationFilePath) {
115:         this.populationFilePath = populationFilePath;
116:     }
117:
118:     public void setRefAnaDate(String refAnaDate) {
119:         this.refAnaDate = refAnaDate;
120:     }
121:
122:     public void saveToFile(File file) throws Exception {
123:         FileOutputStream fos;
124:         ObjectOutputStream oos = null;
125:         try {
126:             fos = new FileOutputStream(file);
127:             oos = new ObjectOutputStream(fos);
128:             oos.writeObject(this);
129:             oos.flush();
130:         } catch (Exception e) {
131:             throw e;
132:         } finally {
133:             if (oos != null) {
134:                 oos.close();
135:             }
136:         }
137:     }
138: }
```

```
136:     }
137: }
138:
139: private static StatisticalEngineConfiguration readFromFile(File file) throws Exception {
140:     FileInputStream fis;
141:     ObjectInputStream ois = null;
142:     try {
143:         fis = new FileInputStream(file);
144:         ois = new ObjectInputStream(fis);
145:         StatisticalEngineConfiguration conf = (StatisticalEngineConfiguration) ois.readObject();
146:         return conf;
147:     } catch (Exception e) {
148:         throw e;
149:     } finally {
150:         if (ois != null) {
151:             ois.close();
152:         }
153:     }
154: }
155:
156: public static StatisticalEngineConfiguration getStatisticalEngineConfiguration() {
157:     if (statisticalEngineConfiguration == null) {
158:         throw new IllegalStateException("Devi caricare prima la configurazione");
159:     }
160:     return statisticalEngineConfiguration;
161: }
162:
163: public static void newStatisticalEngineConfiguration(File f) throws Exception {
164:     if (f.exists()) {
165:         statisticalEngineConfiguration = readFromFile(f);
166:     } else {
167:         statisticalEngineConfiguration = new StatisticalEngineConfiguration();
168:     }
169: }
170: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIROBoxMain.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.main;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.birobox.configuration.StatisticalEngineConfiguration;
34: import eu.biro.birobox.configuration.Configuration;
35: import eu.biro.birobox.configuration.BIROBoxConfiguration;
36: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
37: import eu.biro.birobox.configuration.CommunicationSoftwareConfiguration;
38: import eu.biro.birobox.panel.HomePanel;
39: import eu.biro.birobox.panel.statisticalEngine.StatisticalEnginePanel;
40: import eu.biro.birobox.panel.databaseManager.DatabaseManagerPanel;
41: import eu.biro.birobox.panel.databaseManager.DatabaseConfigurationPanel;
42: import java.awt.event.WindowEvent;
43: import java.io.File;
44: import java.util.logging.Level;
45: import java.util.logging.Logger;
```

```
46: import javax.swing.JFrame;
47: import javax.swing.UIManager;
48: import eu.biro.birobox.panel.adaptor.ConfigurationManagerPanel;
49: import eu.biro.birobox.panel.communicationSoftware.CommunicationSoftwarePanel;
50: import java.awt.event.WindowAdapter;
51:
52:
53: public class BIROBoxMain extends javax.swing.JFrame {
54:
55:     private static BIRODatabaseManagerConfiguration biRODatabaseManagerConfiguration;
56:     private static BIROAdaptorConfigurationList biROAdaptorConfigurationList;
57:     private static BIROBoxConfiguration biROBoxConfiguration;
58:     private static StatisticalEngineConfiguration statisticalEngineConfiguration;
59:     private static CommunicationSoftwareConfiguration communicationSoftwareConfiguration;
60:
61:     /** Creates new form BIROBoxMain */
62:     public BIROBoxMain() {
63:         try {
64:             UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
65:             initComponents();
66:             BIROAdaptorConfigurationList.newBIROAdaptorConfigurationList(new
File(Configuration.bIROAdaptorConfigurationListFilePath));
67:             BIRODatabaseManagerConfiguration.newBIRODatabaseManagerConfiguration(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
68:             BIROBoxConfiguration.newBIROBoxConfiguration(new File(Configuration.bIROBoxConfigurationFilePath));
69:             StatisticalEngineConfiguration.newStatisticalEngineConfiguration(new
File(Configuration.statisticalEngineConfigurationPath));
70:             CommunicationSoftwareConfiguration.newCommunicationSoftwareConfiguration(new
File(Configuration.CommunicationSoftwareConfigurationPath));
71:
72:             this.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
73:             addWindowListener(new WindowAdapter() {
74:
75:                 @Override
76:                 public void windowClosing(WindowEvent e) {
77:                     super.windowClosing(e);
78:                     System.exit(0);
79:                 }
80:             });
81:             rootPanel.addPanel(new HomePanel(rootPanel));
82:         } catch (Exception ex) {
83:             Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
84:         }
85:     }
86:
```

BIROBox/src/eu/biro/birobox/main/BIROBoxMain.java

```
87:  /** This method is called from within the constructor to
88:   * initialize the form.
89:   * WARNING: Do NOT modify this code. The content of this method is
90:   * always regenerated by the Form Editor.
91:   */
92:  // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
93:  private void initComponents() {
94:
95:      buttonPanel = new javax.swing.JPanel();
96:      jXTaskPaneContainer1 = new org.jdesktop.swing.JXTaskPaneContainer();
97:      jXTaskPanel = new org.jdesktop.swing.JXTaskPane();
98:      homeButton = new javax.swing.JButton();
99:      BIROAdaptorTaskPane = new org.jdesktop.swing.JXTaskPane();
100:      configureAdaptorButton = new javax.swing.JButton();
101:      BIRODatabaseTaskPane = new org.jdesktop.swing.JXTaskPane();
102:      configureDatabaseButton = new javax.swing.JButton();
103:      configureDatabaseManagerButton = new javax.swing.JButton();
104:      configureStatisticalEngineButton = new javax.swing.JButton();
105:      jXTaskPane2 = new org.jdesktop.swing.JXTaskPane();
106:      configureCommunicationSoftware = new javax.swing.JButton();
107:      rootPanel = new eu.biro.birobox.panel.RootPanel();
108:      jMenuBar1 = new javax.swing.JMenuBar();
109:      jMenu1 = new javax.swing.JMenu();
110:
111:      setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
112:      setTitle("BIRO Box GUI");
113:
114:      jXTaskPaneContainer1.setBackground(java.awt.SystemColor.activeCaption);
115:
116:      jXTaskPanel.setSpecial(true);
117:      jXTaskPanel.setTitle("Home");
118:
119:      homeButton.setText("Go home");
120:      homeButton.addActionListener(new java.awt.event.ActionListener() {
121:          public void actionPerformed(java.awt.event.ActionEvent evt) {
122:              homeButtonActionPerformed(evt);
123:          }
124:      });
125:      jXTaskPanel1.getContentPane().add(homeButton);
126:
127:      jXTaskPaneContainer1.add(jXTaskPanel);
128:
129:      BIROAdaptorTaskPane.setTitle("BIRO Adaptor");
130:
131:      configureAdaptorButton.setText("Configure & Run Adaptor");
```

BIROBox/src/eu/biro/birobox/main/BIROBoxMain.java

```
132:         configureAdaptorButton.addActionListener(new java.awt.event.ActionListener() {
133:             public void actionPerformed(java.awt.event.ActionEvent evt) {
134:                 configureAdaptorButtonActionPerformed(evt);
135:             }
136:         });
137:         BIROAdaptorTaskPane.getContentPane().add(configureAdaptorButton);
138:
139:         jXTaskPaneContainer1.add(BIROAdaptorTaskPane);
140:
141:         BIRODatabaseTaskPane.setTitle("BIRO Database");
142:
143:         configureDatabaseButton.setText("Configure BIRO Database");
144:         configureDatabaseButton.addActionListener(new java.awt.event.ActionListener() {
145:             public void actionPerformed(java.awt.event.ActionEvent evt) {
146:                 configureDatabaseButtonActionPerformed(evt);
147:             }
148:         });
149:         BIRODatabaseTaskPane.getContentPane().add(configureDatabaseButton);
150:
151:         configureDatabaseManagerButton.setText("Configure & Run Database Manager");
152:         configureDatabaseManagerButton.addActionListener(new java.awt.event.ActionListener() {
153:             public void actionPerformed(java.awt.event.ActionEvent evt) {
154:                 configureDatabaseManagerButtonActionPerformed(evt);
155:             }
156:         });
157:         BIRODatabaseTaskPane.getContentPane().add(configureDatabaseManagerButton);
158:
159:         configureStatisticalEngineButton.setText("Configure & Run Statistical Engine");
160:         configureStatisticalEngineButton.addActionListener(new java.awt.event.ActionListener() {
161:             public void actionPerformed(java.awt.event.ActionEvent evt) {
162:                 configureStatisticalEngineButtonActionPerformed(evt);
163:             }
164:         });
165:         BIRODatabaseTaskPane.getContentPane().add(configureStatisticalEngineButton);
166:
167:         jXTaskPaneContainer1.add(BIRODatabaseTaskPane);
168:
169:         jXTaskPane2.setTitle("BIRO Communication software");
170:
171:         configureCommunicationSoftware.setText("Configure & Run Communicator");
172:         configureCommunicationSoftware.addActionListener(new java.awt.event.ActionListener() {
173:             public void actionPerformed(java.awt.event.ActionEvent evt) {
174:                 configureCommunicationSoftwareActionPerformed(evt);
175:             }
176:         });
```

```
177:         jXTaskPane2.getContentPane().add(configureCommunicationSoftware);
178:
179:         jXTaskPaneContainer1.add(jXTaskPane2);
180:
181:         javax.swing.GroupLayout buttonPanelLayout = new javax.swing.GroupLayout(buttonPanel);
182:         buttonPanel.setLayout(buttonPanelLayout);
183:         buttonPanelLayout.setHorizontalGroup(
184:             buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
185:                 .addComponent(jXTaskPaneContainer1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
186:         );
187:         buttonPanelLayout.setVerticalGroup(
188:             buttonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
189:                 .addComponent(jXTaskPaneContainer1, javax.swing.GroupLayout.DEFAULT_SIZE, 511, Short.MAX_VALUE)
190:         );
191:
192:         rootPanel.setLayout(new java.awt.BorderLayout());
193:
194:         jMenu1.setText("?");
195:         jMenuBar1.add(jMenu1);
196:
197:         setJMenuBar(jMenuBar1);
198:
199:         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
200:         getContentPane().setLayout(layout);
201:         layout.setHorizontalGroup(
202:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
203:                 .addGroup(layout.createSequentialGroup()
204:                     .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
205:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
206:                     .addComponent(rootPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 531, Short.MAX_VALUE))
207:         );
208:         layout.setVerticalGroup(
209:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
210:                 .addGroup(layout.createSequentialGroup()
211:                     .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
212:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
213:                     .addComponent(rootPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 511, Short.MAX_VALUE)
214:                 )
215:         );
216:         pack();
217:     } // </editor-fold> // GEN-END: initComponents
218:     private void configureAdaptorButtonActionPerformed(java.awt.event.ActionEvent evt)
219:     { // GEN-FIRST: event_configureAdaptorButtonActionPerformed
220:         try {
```

BIROBox/src/eu/biro/birobox/main/BIROBoxMain.java

```
218:         rootPanel.addPanel(new ConfigurationManagerPanel(rootPanel));
219:     } catch (Exception ex) {
220:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
221:     }
222: } //GEN-LAST:event_configureAdaptorButtonActionPerformed
223:
224: private void configureCommunicationSoftwareActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_configureCommunicationSoftwareActionPerformed
225:     try {
226:         rootPanel.addPanel(new CommunicationSoftwarePanel(rootPanel));
227:     } catch (Exception ex) {
228:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
229:     }
230: } //GEN-LAST:event_configureCommunicationSoftwareActionPerformed
231:
232: private void configureStatisticalEngineButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_configureStatisticalEngineButtonActionPerformed
233:
234:     try {
235:         rootPanel.addPanel(new StatisticalEnginePanel(rootPanel));
236:     } catch (Exception ex) {
237:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
238:     }
239:
240: } //GEN-LAST:event_configureStatisticalEngineButtonActionPerformed
241:
242: private void configureDatabaseManagerButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_configureDatabaseManagerButtonActionPerformed
243:     try {
244:         rootPanel.addPanel(new DatabaseManagerPanel(rootPanel));
245:     } catch (Exception ex) {
246:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
247:     }
248: } //GEN-LAST:event_configureDatabaseManagerButtonActionPerformed
249:
250: private void configureDatabaseButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_configureDatabaseButtonActionPerformed
251:     try {
252:         rootPanel.addPanel(new DatabaseConfigurationPanel(rootPanel));
253:     } catch (Exception ex) {
254:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
255:     }
256: } //GEN-LAST:event_configureDatabaseButtonActionPerformed
257:
258: private void homeButtonActionPerformed(java.awt.event.ActionEvent evt)
```

```
{//GEN-FIRST:event_homeButtonActionPerformed
259:     try {
260:         rootPanel.addPanel(new HomePanel(rootPanel));
261:     } catch (Exception ex) {
262:         Logger.getLogger(BIROBoxMain.class.getName()).log(Level.SEVERE, null, ex);
263:     }
264:
265: }//GEN-LAST:event_homeButtonActionPerformed
266:
267:     private void saveConfigurationsToFiles() throws Exception {
268:         bIRODatabaseManagerConfiguration.saveToFile(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
269:         bIROAdaptorConfigurationList.saveToFile(new File(Configuration.bIROAdaptorConfigurationListFilePath));
270:         bIROBoxConfiguration.saveToFile(new File(Configuration.bIROBoxConfigurationFilePath));
271:         statisticalEngineConfiguration.saveToFile(new File(Configuration.statisticalEngineConfigurationPath));
272:         communicationSoftwareConfiguration.saveToFile(new
File(Configuration.CommunicationSoftwareConfigurationPath));
273:     }
274:
275:     /**
276:      * @param args the command line arguments
277:      */
278:     public static void main(String args[]) {
279:
280:         System.out.println("root: " + Configuration.root);
281:         System.out.println("java.library.path" + System.getProperty("java.library.path"));
282:
283:         java.awt.EventQueue.invokeLater(new Runnable() {
284:
285:             public void run() {
286:                 BIROBoxMain bIROBoxMain = new BIROBoxMain();
287:                 bIROBoxMain.setLocationRelativeTo(null);
288:                 bIROBoxMain.setVisible(true);
289:             }
290:         });
291:     }
292:     // Variables declaration - do not modify//GEN-BEGIN:variables
293:     private org.jdesktop.swing.JXTaskPane BIROAdaptorTaskPane;
294:     private org.jdesktop.swing.JXTaskPane BIRODatabaseTaskPane;
295:     private javax.swing.JPanel buttonPanel;
296:     private javax.swing.JButton configureAdaptorButton;
297:     private javax.swing.JButton configureCommunicationSoftware;
298:     private javax.swing.JButton configureDatabaseButton;
299:     private javax.swing.JButton configureDatabaseManagerButton;
300:     private javax.swing.JButton configureStatisticalEngineButton;
```

```
301:     private javax.swing.JButton homeButton;
302:     private javax.swing.JMenu jMenu1;
303:     private javax.swing.JMenuBar jMenuBar1;
304:     private org.jdesktop.swingx.JXTaskPane jXTaskPanel1;
305:     private org.jdesktop.swingx.JXTaskPane jXTaskPane2;
306:     private org.jdesktop.swingx.JXTaskPaneContainer jXTaskPaneContainer1;
307:     private eu.biro.birobox.panel.RootPanel rootPanel;
308:     // End of variables declaration//GEN-END:variables
309: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIROFieldTableCellRenderer.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.models;
31:
32: import eu.biro.adaptor2.field.BIROField;
33: import java.awt.Color;
34: import java.awt.Component;
35: import java.awt.Font;
36: import javax.swing.JLabel;
37: import javax.swing.JTable;
38: import javax.swing.table.DefaultTableCellRenderer;
39: import javax.swing.table.TableCellRenderer;
40:
41:
42: public class BIROFieldTableCellRenderer extends DefaultTableCellRenderer {
43:
44:     public BIROFieldTableCellRenderer() {
45:     }
```

```
46:
47:     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean
hasFocus, int row, int column) {
48:         super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
49:         if (value != null) {
50:             BIROField f = (BIROField) value;
51:             setText(f.getName());
52:             if (f.isMandatory()) {
53:                 setFont(new Font("Tahoma", 1, 11));
54:                 setForeground(Color.RED);
55:             } else if (f.isRecorded() && !f.isMandatory()) {
56:                 setFont(new Font("Tahoma", 1, 11));
57:                 setForeground(Color.BLACK);
58:             } else {
59:                 setFont(new Font("Tahoma", 0, 11));
60:                 setForeground(Color.BLACK);
61:             }
62:             setToolTipText(f.getDescription());
63:         }
64:         return this;
65:     }
66: }
67: }
68:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      BIROFieldTableModel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.models;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.adaptor2.BIROAdaptorConfiguration;
34: import eu.biro.adaptor2.field.BIROField;
35: import eu.biro.adaptor2.field.BIROFieldList;
36: import eu.biro.adaptor2.field.BIROFieldType;
37: import java.util.LinkedList;
38: import java.util.List;
39: import javax.swing.table.AbstractTableModel;
40:
41:
42: public class BIROFieldTableModel extends AbstractTableModel {
43:     String columnName = "BIRO field";
44:     List<BIROField> biroFieldList;
45: }
```

```
46:     public BIROFieldTableModel(){
47:         BIROFieldList l =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration().getBIROFieldList();
48:         biroFieldList = new LinkedList<BIROField>();
49:         biroFieldList.addAll(l.getFieldsFor(BIROFieldType.PROFILE));
50:         biroFieldList.addAll(l.getFieldsFor(BIROFieldType.EPISODE));
51:         biroFieldList.addAll(l.getFieldsFor(BIROFieldType.ACTIVITY_DATA));
52:
53:     }
54:
55:     public int getRowCount() {
56:         return biroFieldList.size();
57:     }
58:
59:     public int getColumnCount() {
60:         return 1;
61:     }
62:
63:
64:     public Object getValueAt(int rowIndex, int columnIndex) {
65:         BIROField b = biroFieldList.get(rowIndex);
66:         return b ;
67:     }
68:
69:     @Override
70:     public String getColumnName(int column) {
71:         return columnName;
72:     }
73:
74:     @Override
75:     public Class<?> getColumnClass(int columnIndex) {
76:         return BIROField.class;
77:     }
78:
79:
80:
81: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ConfigurationListModel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10: * the Free Software Foundation; either version 2, or (at your option)
11: * any later version.
12: *
13: * This file is distributed in the hope that it will be useful,
14: * but WITHOUT ANY WARRANTY; without even the implied warranty of
15: * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16: * GNU General Public License for more details.
17: *
18: * You should have received a copy of the GNU General Public License
19: * along with this file; see the file COPYING. If not, write to
20: * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21: *
22: * In short: you may use this file any way you like, as long as you
23: * don't charge money for it, remove this notice, or hold anyone liable
24: * for its results.
25: *
26:
27: * GPL Copyright, The BIRO Project
28: *
29: **/
30:
31: package eu.biro.birobox.models;
32:
33: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
34: import eu.biro.adaptor2.BIROAdaptorConfiguration;
35: import javax.swing.AbstractListModel;
36:
37:
38:
39: public class ConfigurationListModel extends AbstractListModel{
40:
41:     private BIROAdaptorConfigurationList configurationList;
42:
43:     public ConfigurationListModel(BIROAdaptorConfigurationList configurationList) {
44:         this.configurationList= configurationList;
45:     }
```

```
46:
47: public int getSize() {
48:     return configurationList.getConfigurationList().size();
49: }
50:
51: public Object getElementAt(int index) {
52:     return configurationList.getConfigurationList().get(index);
53: }
54:
55: public void addElement(BIROAdaptorConfiguration c) {
56:     configurationList.addBIROAdaptorConfiguration(c);
57:     fireContentsChanged(this, 0, this.getSize());
58: }
59:
60: public void removeElement(BIROAdaptorConfiguration c) {
61:     configurationList.removeConfiguration(c);
62:     fireIntervalRemoved(this, 0, this.getSize());
63: }
64: }
65:
66:
67:
68:
69: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:          DBMSDriverComboBoxModel.java
5:  * Authors:       Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * donât charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.models;
31:
32: import eu.biro.adaptor2.DBMSDriver;
33: import java.util.Vector;
34: import javax.swing.DefaultComboBoxModel;
35:
36:
37: public class DBMSDriverComboBoxModel extends DefaultComboBoxModel {
38:
39:
40:
41:     public DBMSDriverComboBoxModel(Vector<DBMSDriver> v) {
42:         super(v);
43:     }
44: }
45:
```

```
46: public void setSelectedDriver(DBMSDriver dbMSdriver) {
47:     DBMSDriver candidateDriver = null;
48:     DBMSDriver examinedDriver;
49:     for (int i = 0; i < super.getSize(); i++) {
50:         examinedDriver = (DBMSDriver) super.elementAt(i);
51:         if (examinedDriver.compare(dbMSdriver)) {
52:             candidateDriver = examinedDriver;
53:         }
54:     }
55:     if (candidateDriver != null) {
56:         super.setSelectedItem(candidateDriver);
57:     }
58:
59: }
60: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:          StaticFieldTableCellRenderer.java
5:  * Authors:       Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.models;
31:
32: import eu.biro.adaptor2.field.StaticBIROField;
33: import java.awt.Component;
34: import javax.swing.JComboBox;
35: import javax.swing.JTable;
36: import javax.swing.table.DefaultTableCellRenderer;
37:
38:
39: public class StaticFieldTableCellRenderer extends DefaultTableCellRenderer {
40:
41:     public StaticFieldTableCellRenderer() {
42:     }
43:
44:     @Override
45:     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean
```

```
hasFocus, int row, int column) {
46:     super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
47:     if (value != null) {
48:         StaticBIROField f = (StaticBIROField) value;
49:         if (column == 0) {
50:             if (f.isMandatory()) {
51:                 setText(f.getName() + " *");
52:             } else {
53:                 setText(f.getName());
54:             }
55:             setToolTipText(f.getDescription());
56:         } else if (column == 1) {
57:             if (f.getValue() != null) {
58:                 setText(f.getValue().toString());
59:             } else {
60:                 setText("");
61:             }
62:         }
63:     }
64: }
65: return this;
66:
67: }
68: }
69: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:          StaticFieldTableModel.java
5:  * Authors:       Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.models;
32:
33: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
34: import eu.biro.adaptor2.field.BIROField;
35: import eu.biro.adaptor2.field.BIROFieldList;
36: import eu.biro.adaptor2.field.BIROFieldType;
37: import eu.biro.adaptor2.field.StaticBIROField;
38: import java.util.LinkedList;
39: import java.util.List;
40: import javax.swing.JOptionPane;
41: import javax.swing.table.AbstractTableModel;
42:
43:
44: public class StaticFieldTableModel extends AbstractTableModel {
45:
```

```
46:     String columnName[] = {"Field", "Value"};
47:     List<BIROField> biroFieldList;
48:     BIROFieldType type;
49:
50:     public StaticFieldTableModel(BIROFieldType type) {
51:         this.type = type;
52:         BIROFieldList l =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration().getBIROFieldList();
53:         biroFieldList = new LinkedList<BIROField>();
54:         biroFieldList.addAll(l.getFieldsFor(type));
55:         biroFieldList.remove(l.getFieldWithName("DS_ID"));
56:         biroFieldList.remove(l.getFieldWithName("DS_NAME"));
57:         biroFieldList.remove(l.getFieldWithName("DS_TYPE"));
58:     }
59:
60:     public int getRowCount() {
61:         return biroFieldList.size();
62:     }
63:
64:     public int getColumnCount() {
65:         return 2;
66:     }
67:
68:
69:     public Object getValueAt(int rowIndex, int columnIndex) {
70:         BIROField f = biroFieldList.get(rowIndex);
71:         Object o = null;
72:         if (columnIndex == 0) {
73:             o = f;
74:         } else {
75:             o = ((StaticBIROField) f).getValue();
76:         }
77:
78:         return o;
79:     }
80:
81:     @Override
82:     public String getColumnName(int column) {
83:         return columnName[column];
84:     }
85:
86:     @Override
87:     public Class<?> getColumnClass(int columnIndex) {
88:         return columnIndex == 0 ? BIROField.class : type == BIROFieldType.SITE_HEADER ? String.class : Integer.
class;
```

```
89:     }
90:
91:     @Override
92:     public boolean isCellEditable(int rowIndex, int columnIndex) {
93:         return columnIndex == 1;
94:     }
95:
96:
97:     @Override
98:     public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
99:         try {
100:             if (columnIndex == 1) {
101:                 BIROField b = biroFieldList.get(rowIndex);
102:                 ((StaticBIROField) b).setValue(aValue);
103:             }
104:         } catch (Exception e) {
105:             JOptionPane.showMessageDialog(null, e);
106:         }
107:     }
108: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalObjectTableCellEditor.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.models;
32:
33: import java.awt.Component;
34: import java.awt.event.ItemEvent;
35: import java.awt.event.ItemListener;
36: import javax.swing.DefaultCellEditor;
37: import javax.swing.JCheckBox;
38: import javax.swing.JRadioButton;
39: import javax.swing.JTable;
40:
41:
42: public class StatisticalObjectTableCellEditor extends DefaultCellEditor implements ItemListener {
43:     private JRadioButton button;
44:
45:     public StatisticalObjectTableCellEditor(JCheckBox checkBox) {
```

```
46:     super(checkBox);
47: }
48:
49:     @Override
50:     public Component getTableCellEditorComponent(JTable table, Object value, boolean isSelected, int row, int column)
{
51:         if (value == null)
52:             return null;
53:         button = (JRadioButton) value;
54:         button.addItemListener(this);
55:         return (Component) value;
56:     }
57:
58:     @Override
59:     public Object getCellEditorValue() {
60:         button.removeItemListener(this);
61:         return button;
62:     }
63:
64:     public void itemStateChanged(ItemEvent e) {
65:         super.fireEditingStopped();
66:     }
67: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalObjectTableCellRenderer.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.models;
32:
33: import eu.biro.birobox.panel.communicationSoftware.StatisticalObject;
34: import java.awt.Component;
35: import java.sql.Timestamp;
36: import javax.swing.JTable;
37: import javax.swing.table.DefaultTableCellRenderer;
38:
39:
40: public class StatisticalObjectTableCellRenderer extends DefaultTableCellRenderer {
41:
42:     public StatisticalObjectTableCellRenderer() {
43:     }
44:
45:     @Override
```

```
46:     public Component getTableCellRendererComponent(JTable table, Object value, boolean isSelected, boolean
hasFocus, int row, int column) {
47:         super.getTableCellRendererComponent(table, value, isSelected, hasFocus, row, column);
48:         Object o;
49:         if (value != null) {
50:             StatisticalObject s = (StatisticalObject) value;
51:
52:             if (column == 0) {
53:                 setText(s.toString());
54:             } else if (column == 1) {
55:                 setText(s.getCreationTimestamp().toString());
56:             } else if (column == 2) {
57:                 int i = s.getSendingTimestamps().size();
58:                 if (i != 0) {
59:                     setText(s.getSendingTimestamps().elementAt(i-1).toString());
60:                 } else {
61:                     setText("empty");
62:                 }
63:             }
64:         }
65:         return this;
66:
67:     }
68:
69:
70: }
71:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalObjectTableModel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31:
32: package eu.biro.birobox.models;
33:
34: import eu.biro.birobox.configuration.CommunicationSoftwareConfiguration;
35: import eu.biro.birobox.panel.communicationSoftware.StatisticalObject;
36: import java.util.Vector;
37: import javax.swing.table.AbstractTableModel;
38:
39: public class StatisticalObjectTableModel extends AbstractTableModel {
40:
41:     String columnName[] = {"id", "creation date", "last sending date", "send"};
42:     Vector<StatisticalObject> statisticalObjects;
43:
44:
45:     public StatisticalObjectTableModel() {
```

```
46:         this.statisticalObjects =
(CommunicationSoftwareConfiguration.getCommunicationSoftwareConfiguration()).getStatisticalObjects();
47:     }
48:
49:     @Override
50:     public int getRowCount() {
51:         return statisticalObjects.size();
52:     }
53:
54:     @Override
55:     public int getColumnCount() {
56:         return 3;
57:     }
58:
59:     @Override
60:     public Object getValueAt(int rowIndex, int columnIndex) {
61:         StatisticalObject s = statisticalObjects.get(rowIndex);
62:         return s;
63:     }
64:
65:     @Override
66:     public String getColumnName(int column) {
67:         return columnName[column];
68:     }
69:
70:     @Override
71:     public Class<?> getColumnClass(int columnIndex) {
72:         return StatisticalObject.class;
73:         //return columnIndex == 0 ? String.class : columnIndex == 1 ? String.class : columnIndex == 2 ? String.
class : JRadioButton.class;
74:     }
75:
76:     @Override
77:     public boolean isCellEditable(int rowIndex, int columnIndex) {
78:         return false;
79:     }
80:
81:
82:     @Override
83:     public void setValueAt(Object aValue, int rowIndex, int columnIndex) {
84:         fireTableCellUpdated(rowIndex, columnIndex);
85:         fireTableDataChanged();
86:
87:     }
88: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ChildrenPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.panel;
32:
33: import javax.swing.JPanel;
34:
35: public abstract class ChildrenPanel extends JPanel {
36:     public abstract void saveData() throws Exception;
37:
38: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      HomePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel;
31:
32: import eu.biro.birobox.panel.ChildrenPanel;
33:
34:
35: public class HomePanel extends ChildrenPanel {
36:
37:     private RootPanel rootPanel;
38:
39:     /** Creates new form HomePanel */
40:     public HomePanel(RootPanel rootPanel) {
41:         this.rootPanel = rootPanel;
42:         initComponents();
43:     }
44: }
45:
```

```

46:     /** This method is called from within the constructor to
47:      * initialize the form.
48:      * WARNING: Do NOT modify this code. The content of this method is
49:      * always regenerated by the Form Editor.
50:      */
51:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN:initComponents
52:     private void initComponents() {
53:
54:         jLabel1 = new javax.swing.JLabel();
55:
56:         setBorder(javax.swing.BorderFactory.createTitledBorder("Home"));
57:
58:         jLabel1.setIcon(new javax.swing.ImageIcon(getClass().getResource("/biro_logo.jpg"))); // NOI18N
59:         jLabel1.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
60:
61:         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
62:         this.setLayout(layout);
63:         layout.setHorizontalGroup(
64:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
65:                 .addGroup(layout.createSequentialGroup()
66:                     .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
67:                     .addGap(298, 298, 298))
68:                 .addGroup(layout.createSequentialGroup()
69:                     .setVerticalGroup(
70:                         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
71:                             .addGroup(layout.createSequentialGroup()
72:                                 .addComponent(jLabel1)
73:                                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
74:                                 .addContainerGap(142, Short.MAX_VALUE))
75:                             .addContainerGap())
76:                     .addContainerGap())
77:         );
78:     }
79:     }
80:     // Variables declaration - do not modify//GEN-BEGIN:variables
81:     private javax.swing.JLabel jLabel1;
82:     // End of variables declaration//GEN-END:variables
83: }

```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      RootPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * do not charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel;
31:
32: import eu.biro.birobox.panel.ChildrenPanel;
33: import java.awt.BorderLayout;
34: import java.io.IOException;
35: import javax.swing.JPanel;
36:
37: public class RootPanel extends JPanel {
38:
39:     private ChildrenPanel childrenPanel;
40:
41:     public RootPanel() {
42:         super();
43:     }
44:
45:     public void addPanel(ChildrenPanel childrenPanel) throws Exception {
```

```
46:         savePanel();
47:         removeAll();
48:         this.childrenPanel = childrenPanel;
49:         add(childrenPanel, BorderLayout.CENTER);
50:         revalidate();
51:         repaint();
52:     }
53:
54:     public void savePanel() throws Exception {
55:         if (childrenPanel != null) {
56:             childrenPanel.saveData();
57:         }
58:     }
59: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ActivityTableConfigurationPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.utils.ConnectionManager;
33: import eu.biro.birobox.panel.*;
34: import eu.biro.birobox.main.*;
35: import eu.biro.birobox.panel.ChildrenPanel;
36: import eu.biro.birobox.panel.adaptor.DataSourceConfigurationPanel;
37: import eu.biro.adaptor2.BIROAdaptorConfiguration;
38: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
39: import java.io.IOException;
40: import java.sql.DatabaseMetaData;
41: import java.sql.ResultSet;
42: import java.sql.SQLException;
43: import java.util.Vector;
44: import java.util.logging.Level;
45: import java.util.logging.Logger;
```

```
46: import javax.swing.JOptionPane;
47:
48:
49: public class ActivityTableConfigurationPanel extends ChildrenPanel {
50:
51:     private RootPanel rootPanel;
52:     BIROAdaptorConfiguration biROAdaptorConfiguration;
53:     private ConnectionManager connectionManager;
54:
55:     /** Creates new form NewJPanel */
56:     public ActivityTableConfigurationPanel(RootPanel rootPanel) throws IOException {
57:         this.rootPanel = rootPanel;
58:         connectionManager = ConnectionManager.getManager();
59:         initComponents();
60:         loadData();
61:     }
62:
63:     /** This method is called from within the constructor to
64:      * initialize the form.
65:      * WARNING: Do NOT modify this code. The content of this method is
66:      * always regenerated by the Form Editor.
67:      */
68:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
69:     private void initComponents() {
70:
71:         buttonGroup1 = new javax.swing.ButtonGroup();
72:         ActivityTableConfigurationPanel = new javax.swing.JPanel();
73:         descriptionLabel = new javax.swing.JLabel();
74:         buttonPanel = new javax.swing.JPanel();
75:         previousButton = new javax.swing.JButton();
76:         nextButton = new javax.swing.JButton();
77:         activityTableCreationRadioButton = new javax.swing.JRadioButton();
78:         activityTableSelectionRadioButton = new javax.swing.JRadioButton();
79:         jScrollPane1 = new javax.swing.JScrollPane();
80:         activityTableCreationTextArea = new javax.swing.JTextArea();
81:         activityTableSelectionButton = new javax.swing.JButton();
82:         activityTableSelectionTextField = new javax.swing.JTextField();
83:         activityTableEnableRadioButton = new javax.swing.JRadioButton();
84:
85:         ActivityTableConfigurationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Activitytable
configuration"));
86:
87:         descriptionLabel.setText("Configure the activitytable");
88:
89:         buttonPanel.setMinimumSize(new java.awt.Dimension(0, 0));
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ActivityTableConfigurationPanel.java

```
90:         buttonPanel.setLayout(new java.awt.GridLayout(1, 2, 5, 0));
91:
92:         previousButton.setText("Previous");
93:         previousButton.addActionListener(new java.awt.event.ActionListener() {
94:             public void actionPerformed(java.awt.event.ActionEvent evt) {
95:                 previousButtonActionPerformed(evt);
96:             }
97:         });
98:         buttonPanel.add(previousButton);
99:
100:        nextButton.setText("Next");
101:        nextButton.addActionListener(new java.awt.event.ActionListener() {
102:            public void actionPerformed(java.awt.event.ActionEvent evt) {
103:                nextButtonActionPerformed(evt);
104:            }
105:        });
106:        buttonPanel.add(nextButton);
107:
108:        buttonGroup1.add(activityTableCreationRadioButton);
109:        activityTableCreationRadioButton.setText("select the activitytable from a custom query");
110:        activityTableCreationRadioButton.addChangeListener(new javax.swing.event.ChangeListener() {
111:            public void stateChanged(javax.swing.event.ChangeEvent evt) {
112:                activityTableCreationRadioButtonStateChanged(evt);
113:            }
114:        });
115:
116:        buttonGroup1.add(activityTableSelectionRadioButton);
117:        activityTableSelectionRadioButton.setSelected(true);
118:        activityTableSelectionRadioButton.setText("select the activitytable from already present table");
119:        activityTableSelectionRadioButton.addChangeListener(new javax.swing.event.ChangeListener() {
120:            public void stateChanged(javax.swing.event.ChangeEvent evt) {
121:                activityTableSelectionRadioButtonStateChanged(evt);
122:            }
123:        });
124:
125:        activityTableCreationTextArea.setColumns(20);
126:        activityTableCreationTextArea.setLineWrap(true);
127:        activityTableCreationTextArea.setRows(5);
128:        activityTableCreationTextArea.setEnabled(false);
129:        activityTableCreationTextArea.setOpaque(false);
130:        jScrollPane1.setViewportView(activityTableCreationTextArea);
131:
132:        activityTableSelectionButton.setText("...");
133:        activityTableSelectionButton.addActionListener(new java.awt.event.ActionListener() {
134:            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

```
135:         activityTableSelectionButtonActionPerformed(evt);
136:     }
137: });
138:
139:     buttonGroup1.add(activityTableEnableRadioButton);
140:     activityTableEnableRadioButton.setText("don't select the activitytable");
141:     activityTableEnableRadioButton.setAutoscrolls(true);
142:
143:     javax.swing.GroupLayout ActivityTableConfigurationPanelLayout = new
javax.swing.GroupLayout(ActivityTableConfigurationPanel);
144:     ActivityTableConfigurationPanel.setLayout(ActivityTableConfigurationPanelLayout);
145:     ActivityTableConfigurationPanelLayout.setHorizontalGroup(
146:         ActivityTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
147:         .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup())
148:         .addGroup(ActivityTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
149:             .addComponent(descriptionLabel)
150:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
ActivityTableConfigurationPanelLayout.createSequentialGroup())
151:             .addGroup(ActivityTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
152:                 .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 525, Short.MAX_VALUE)
153:                 .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup())
154:                 .addComponent(activityTableSelectionTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 464, Short.MAX_VALUE)
155:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
156:                 .addComponent(activityTableSelectionButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 55, javax.swing.GroupLayout.PREFERRED_SIZE)))
157:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
158:         .addContainerGap())
159:         .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 525, Short.MAX_VALUE)
160:         .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup())
161:         .addComponent(activityTableCreationRadioButton, javax.swing.GroupLayout.PREFERRED_SIZE, 350,
javax.swing.GroupLayout.PREFERRED_SIZE)
162:         .addContainerGap())
163:         .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup())
164:         .addComponent(activityTableSelectionRadioButton, javax.swing.GroupLayout.PREFERRED_SIZE, 356,
javax.swing.GroupLayout.PREFERRED_SIZE)
165:         .addContainerGap(169, Short.MAX_VALUE))
166:         .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup())
167:         .addComponent(activityTableEnableRadioButton)
168:         .addContainerGap())
169:     );
170:     ActivityTableConfigurationPanelLayout.setVerticalGroup(
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ActivityTableConfigurationPanel.java

```
171:         ActivityTableConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
172:         .addGroup(ActivityTableConfigurationPanelLayout.createSequentialGroup( )
173:         .addComponent(descriptionLabel)
174:         .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
175:         .addComponent(activityTableSelectionRadioButton)
176:         .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
177:
.addGroup(ActivityTableConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
178:         .addComponent(activityTableSelectionButton)
179:         .addComponent(activityTableSelectionTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
180:         .addGap(18, 18, 18)
181:         .addComponent(activityTableCreationRadioButton)
182:         .addGap(7, 7, 7)
183:         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
184:         .addGap(18, 18, 18)
185:         .addComponent(activityTableEnableRadioButton)
186:         .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED, 260, Short.MAX_VALUE)
187:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
188:     );
189:
190:     ActivityTableConfigurationPanelLayout.linkSize( javax.swing.SwingConstants.VERTICAL, new
java.awt.Component[] {activityTableSelectionButton, activityTableSelectionTextField});
191:
192:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
193:     this.setLayout(layout);
194:     layout.setHorizontalGroup(
195:         layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
196:         .addComponent(ActivityTableConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
197:     );
198:     layout.setVerticalGroup(
199:         layout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
200:         .addComponent(ActivityTableConfigurationPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
201:     );
202: }// </editor-fold>//GEN-END:initComponents
203: private void nextButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_nextButtonActionPerformed
204:     try {
205:         saveData();
206:         checkConnection();
207:         rootPanel.addPanel(new DataSourceConfigurationPanel(rootPanel));
```

```
208:         } catch (Exception ex) {
209:             JOptionPane.showMessageDialog(this, ex.getMessage(), "WARNING", JOptionPane.WARNING_MESSAGE);
210:             Logger.getLogger(ActivityTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
211:         }
212:     } //GEN-LAST:event_nextButtonActionPerformed
213:
214:     private void activityTableCreationRadioButtonStateChanged(javax.swing.event.ChangeEvent evt)
215: { //GEN-FIRST:event_activityTableCreationRadioButtonStateChanged
216:         activityTableCreationTextArea.setEnabled(activityTableCreationRadioButton.isSelected());
217:         activityTableCreationTextArea.setOpaque(activityTableCreationRadioButton.isSelected());
218:     } //GEN-LAST:event_activityTableCreationRadioButtonStateChanged
219:
220:     private void activityTableSelectionRadioButtonStateChanged(javax.swing.event.ChangeEvent evt)
221: { //GEN-FIRST:event_activityTableSelectionRadioButtonStateChanged
222:         activityTableSelectionTextField.setEnabled(activityTableSelectionRadioButton.isSelected());
223:         activityTableSelectionButton.setEnabled(activityTableSelectionRadioButton.isSelected());
224:
225:     } //GEN-LAST:event_activityTableSelectionRadioButtonStateChanged
226:
227:     private void activityTableSelectionButtonActionPerformed(java.awt.event.ActionEvent evt)
228: { //GEN-FIRST:event_activityTableSelectionButtonActionPerformed
229:         try {
230:             Object[] values = getTablesList().toArray();
231:             Object value = values[0];
232:             String s = (String) JOptionPane.showInputDialog(this, "Please, select a table", "ActivityData Table
name selection", JOptionPane.PLAIN_MESSAGE, null, values, value);
233:             if (s != null) {
234:                 activityTableSelectionTextField.setText(s);
235:             }
236:         } catch (Exception ex) {
237:             JOptionPane.showMessageDialog(this, ex.getMessage(), "Warning", JOptionPane.WARNING_MESSAGE);
238:             Logger.getLogger(ActivityTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
239:         }
240:     } //GEN-LAST:event_activityTableSelectionButtonActionPerformed
241:
242:     private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
243: { //GEN-FIRST:event_previousButtonActionPerformed
244:         try {
245:             rootPanel.addPanel(new MergeTableConfigurationPanel(rootPanel));
246:         } catch (Exception ex) {
247:             Logger.getLogger(ActivityTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
248:         }
249:     }
```

```
248:     } //GEN-LAST:event_previousButtonActionPerformed
249:
250:     private void loadData() throws IOException {
251:         bIROAdaptorConfiguration =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration();
252:         activityTableSelectionRadioButton.setSelected(bIROAdaptorConfiguration.isActivityTableSelected() &&
bIROAdaptorConfiguration.isActivityTableEnabled());
253:         activityTableCreationRadioButton.setSelected(!bIROAdaptorConfiguration.isActivityTableSelected() &&
bIROAdaptorConfiguration.isActivityTableEnabled());
254:         activityTableEnableRadioButton.setSelected(!bIROAdaptorConfiguration.isActivityTableEnabled());
255:         activityTableSelectionTextField.setText(bIROAdaptorConfiguration.getActivityTableName());
256:         activityTableCreationTextArea.setText(bIROAdaptorConfiguration.getActivityTableCreationQuery());
257:     }
258:
259:     public void saveData() throws IOException {
260:         bIROAdaptorConfiguration.setActivityTableSelected(activityTableSelectionRadioButton.isSelected());
261:         bIROAdaptorConfiguration.setActivityTableName(activityTableSelectionTextField.getText());
262:         bIROAdaptorConfiguration.setActivityTableCreationQuery(activityTableCreationTextArea.getText());
263:         bIROAdaptorConfiguration.setActivityTableEnabled(!activityTableEnableRadioButton.isSelected());
264:
265:         if (activityTableSelectionRadioButton.isSelected()) {
266:             String activityTableQuery = "SELECT * FROM \"" + activityTableSelectionTextField.getText()+"\"";
267:             bIROAdaptorConfiguration.setActivityTableQuery(activityTableQuery);
268:             System.out.println(bIROAdaptorConfiguration.getActivityTableQuery());
269:         } else if (activityTableCreationRadioButton.isSelected()) {
270:             bIROAdaptorConfiguration.setActivityTableQuery(activityTableCreationTextArea.getText());
271:         } else {
272:             bIROAdaptorConfiguration.setActivityTableQuery("");
273:         }
274:
275:     }
276:
277:     private Vector<String> getTablesList() throws SQLException, ClassNotFoundException, Exception {
278:         Vector<String> tables = new Vector<String>();
279:         connectionManager.connect();
280:         DatabaseMetaData md = connectionManager.getMetaData();
281:
282:         ResultSet rs = md.getTables(null, null, "%", new String[]{"TABLE"});
283:         while (rs.next()) {
284:             tables.add(rs.getString(3));
285:         }
286:         System.out.println(md.getDatabaseProductName());
287:         System.out.println(md.getDriverName());
288:
289:         connectionManager.close();
```

```
290:         return tables;
291:
292:     }
293:
294:     private void checkConnection() throws Exception {
295:         if (!activityTableEnableRadioButton.isSelected()) {
296:             try {
297:                 ConnectionManager manager =
ConnectionManager.initializeManager(bIROAdaptorConfiguration.getDBMSDriver());
298:                 manager.connect();
299:                 ResultSet r = manager.executeQuery(bIROAdaptorConfiguration.getActivityTableQuery());
300:                 if (!r.next()) {
301:                     throw new Exception("The query returned an empty result set");
302:                 }
303:             } catch (Exception ex) {
304:                 Logger.getLogger(ActivityTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
305:                 throw new Exception("An error occurred when attempting to establish a connection to the merge
table.\n" + ex.getMessage());
306:             }
307:         }
308:     }
309:     // Variables declaration - do not modify//GEN-BEGIN:variables
310:     private javax.swing.JPanel ActivityTableConfigurationPanel;
311:     private javax.swing.JRadioButton activityTableCreationRadioButton;
312:     private javax.swing.JTextArea activityTableCreationTextArea;
313:     private javax.swing.JRadioButton activityTableEnableRadioButton;
314:     private javax.swing.JButton activityTableSelectionButton;
315:     private javax.swing.JRadioButton activityTableSelectionRadioButton;
316:     private javax.swing.JTextField activityTableSelectionTextField;
317:     private javax.swing.ButtonGroup buttonGroup1;
318:     private javax.swing.JPanel buttonPanel;
319:     private javax.swing.JLabel descriptionLabel;
320:     private javax.swing.JScrollPane jScrollPane1;
321:     private javax.swing.JButton nextButton;
322:     private javax.swing.JButton previousButton;
323:     // End of variables declaration//GEN-END:variables
324: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      AdaptorProgressPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.adaptor2.AdaptorProgressListener;
33: import eu.biro.adaptor2.exporter.BIROAdaptor2;
34: import javax.swing.JComponent;
35: import javax.swing.JFrame;
36: import javax.swing.SwingUtilities;
37:
38:
39: public class AdaptorProgressPanel extends javax.swing.JPanel implements AdaptorProgressListener {
40:
41:     /** Creates new form AdaptorProgressPanel */
42:     public AdaptorProgressPanel() {
43:         initComponents();
44:     }
45: }
```

```
46:  /** This method is called from within the constructor to
47:   * initialize the form.
48:   * WARNING: Do NOT modify this code. The content of this method is
49:   * always regenerated by the Form Editor.
50:   */
51:  // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
52:  private void initComponents() {
53:
54:      adaptorProgressBar = new javax.swing.JProgressBar();
55:      AdaptorStatusLabel = new javax.swing.JLabel();
56:      jScrollPane1 = new javax.swing.JScrollPane();
57:      outputTextArea = new javax.swing.JTextArea();
58:
59:      AdaptorStatusLabel.setText("BIRO Adaptor Progress status:");
60:
61:      outputTextArea.setColumns(20);
62:      outputTextArea.setEditable(false);
63:      outputTextArea.setRows(5);
64:      jScrollPane1.setViewportView(outputTextArea);
65:
66:      javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
67:      this.setLayout(layout);
68:      layout.setHorizontalGroup(
69:          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
70:              .addGroup(layout.createSequentialGroup()
71:                  .addComponent(jScrollPane1)
72:                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
73:                      .addComponent(AdaptorStatusLabel)
74:                      .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 301, Short.MAX_VALUE)
75:                      .addComponent(adaptorProgressBar, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 301, Short.MAX_VALUE))
76:                  .addGap())
77:      );
78:      layout.setVerticalGroup(
79:          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
80:              .addGroup(layout.createSequentialGroup()
81:                  .addComponent(jScrollPane1)
82:                  .addComponent(AdaptorStatusLabel)
83:                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
84:                  .addComponent(adaptorProgressBar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
85:                  .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
86:                  .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 98, Short.MAX_VALUE)
87:                  .addGap())
88:      );
```

```
89:  }// </editor-fold>//GEN-END:initComponents
90:  // Variables declaration - do not modify//GEN-BEGIN:variables
91:  private javax.swing.JLabel AdaptorStatusLabel;
92:  private javax.swing.JProgressBar adaptorProgressBar;
93:  private javax.swing.JScrollPane jScrollPane1;
94:  private javax.swing.JTextArea outputTextArea;
95:  // End of variables declaration//GEN-END:variables
96:  public void setMaximumForProgress(final int max) {
97:      SwingUtilities.invokeLater(new Runnable() {
98:
99:          public void run() {
100:              adaptorProgressBar.setMaximum(max);
101:          }
102:      });
103:
104:  }
105:
106:  public void showText(String txt) {
107:      final String t = txt;
108:      SwingUtilities.invokeLater(new Runnable() {
109:
110:          public void run() {
111:              outputTextArea.append(t + "\n");
112:          }
113:      });
114:  }
115:
116:  public void updateProgressValue() {
117:      SwingUtilities.invokeLater(new Runnable() {
118:
119:          public void run() {
120:              adaptorProgressBar.setValue(adaptorProgressBar.getValue() + 1);
121:              AdaptorStatusLabel.setText(String.format("Adaptor progress status: %d%% completed",
122: ((adaptorProgressBar.getValue() * 100) / adaptorProgressBar.getMaximum())));
123:          }
124:      });
125:  }
126:  /**
127:   * Create the GUI and show it. As with all GUI code, this must run
128:   * on the event-dispatching thread.
129:   */
130:  public static AdaptorProgressPanel createAndShowAdaptorProgress(final BIROAdaptor2 ad) {
131:
132:      //Create and set up the content pane.
```

```
133:     final AdaptorProgressPanel progressPanel = new AdaptorProgressPanel();
134:     javax.swing.SwingUtilities.invokeLater(new Runnable() {
135:
136:         public void run() {
137:             //Create and set up the window.
138:             JFrame frame = new JFrame("Progress Status");
139:             frame.setResizable(true);
140:
141:
142:             JComponent newContentPane = progressPanel;
143:             newContentPane.setOpaque(true); //content panes must be opaque
144:             frame.setContentPane(newContentPane);
145:             frame.setLocationRelativeTo(null);
146:             ad.addProgressListener(progressPanel);
147:
148:
149:             //Display the window.
150:             frame.pack();
151:             frame.setVisible(true);
152:         }
153:     });
154:     return progressPanel;
155:
156:
157: }
158: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ConfigurationManagerPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.adaptor2.BIROAdaptorConfiguration;
34: import eu.biro.adaptor2.exporter.BIROAdaptor2;
35: import eu.biro.adaptor2.field.BIROField;
36: import eu.biro.birobox.configuration.BIROBoxConfiguration;
37: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
38: import eu.biro.birobox.configuration.Configuration;
39: import java.awt.Cursor;
40: import java.io.IOException;
41: import java.sql.SQLException;
42: import javax.swing.JOptionPane;
43: import javax.swing.JTextField;
44: import javax.swing.event.ListSelectionEvent;
45: import eu.biro.birobox.panel.*;
```

```
46: import java.util.logging.Level;
47: import java.util.logging.Logger;
48: import javax.swing.event.ListSelectionListener;
49: import eu.biro.birobox.models.ConfigurationListModel;
50: import eu.biro.birobox.utils.ConnectionManager;
51: import java.io.File;
52: import java.sql.ResultSet;
53:
54:
55: public class ConfigurationManagerPanel extends ChildrenPanel {
56:
57:     private RootPanel rootPanel;
58:     private BIROAdaptorConfigurationList bIROAdaptorConfigurationList;
59:     private BIROBoxConfiguration bIROBoxConfiguration;
60:     private BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
61:
62:     /** Creates new form ConfigurationManagerPanel */
63:     public ConfigurationManagerPanel(RootPanel rootPanel) throws IOException {
64:         this.rootPanel = rootPanel;
65:         bIROAdaptorConfigurationList = BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList();
66:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
67:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
68:
69:         initComponents();
70:
71:         configurationList.addListSelectionListener(new ListSelectionListener() {
72:
73:             public void valueChanged(ListSelectionEvent e) {
74:                 if (e.getValueIsAdjusting() == false) {
75:
76:                     if (configurationList.getSelectedIndex() == -1) {
77:                         //No selection, disable fire button.
78:                         copyButton.setEnabled(false);
79:                         editButton.setEnabled(false);
80:                         deleteButton.setEnabled(false);
81:                         saveAndRunButton.setEnabled(false);
82:
83:                     } else {
84:                         //Selection, enable the fire button.
85:                         copyButton.setEnabled(true);
86:                         editButton.setEnabled(true);
87:                         deleteButton.setEnabled(true);
88:                         saveAndRunButton.setEnabled(true);
89:                     }
90:                 }
91:             }
92:         });
93:     }
94: }
```

```
91:         }
92:     });
93:     loadData();
94: }
95:
96: /** This method is called from within the constructor to
97:  * initialize the form.
98:  * WARNING: Do NOT modify this code. The content of this method is
99:  * always regenerated by the Form Editor.
100:  */
101: // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
102: private void initComponents() {
103:
104:     newButton = new javax.swing.JButton();
105:     copyButton = new javax.swing.JButton();
106:     deleteButton = new javax.swing.JButton();
107:     jLabel1 = new javax.swing.JLabel();
108:     jPanel1 = new javax.swing.JPanel();
109:     jPanel2 = new javax.swing.JPanel();
110:     saveAndRunButton = new javax.swing.JButton();
111:     jPanel3 = new javax.swing.JPanel();
112:     saveAndExitButton = new javax.swing.JButton();
113:     jScrollPane1 = new javax.swing.JScrollPane();
114:     configurationList = new javax.swing.JList();
115:     editButton = new javax.swing.JButton();
116:
117:     setBorder(javax.swing.BorderFactory.createTitledBorder("Adaptor configuration manager"));
118:
119:     newButton.setText("New");
120:     newButton.addActionListener(new java.awt.event.ActionListener() {
121:         public void actionPerformed(java.awt.event.ActionEvent evt) {
122:             newButtonActionPerformed(evt);
123:         }
124:     });
125:
126:     copyButton.setText("Copy");
127:     copyButton.addActionListener(new java.awt.event.ActionListener() {
128:         public void actionPerformed(java.awt.event.ActionEvent evt) {
129:             copyButtonActionPerformed(evt);
130:         }
131:     });
132:
133:     deleteButton.setText("Delete");
134:     deleteButton.addActionListener(new java.awt.event.ActionListener() {
135:         public void actionPerformed(java.awt.event.ActionEvent evt) {
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
136:         deleteButtonActionPerformed(evt);
137:     }
138: });
139:
140: jLabel1.setText("Select Adaptor configuration:");
141:
142: jPanel1.setLayout(new java.awt.GridLayout(1, 2, 5, 5));
143:
144: saveAndRunButton.setText("Save and run Adaptor");
145: saveAndRunButton.addActionListener(new java.awt.event.ActionListener() {
146:     public void actionPerformed(java.awt.event.ActionEvent evt) {
147:         saveAndRunButtonActionPerformed(evt);
148:     }
149: });
150:
151: javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
152: jPanel2.setLayout(jPanel2Layout);
153: jPanel2Layout.setHorizontalGroup(
154:     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
155:         .addComponent(saveAndRunButton, javax.swing.GroupLayout.DEFAULT_SIZE, 222, Short.MAX_VALUE)
156: );
157: jPanel2Layout.setVerticalGroup(
158:     jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
159:         .addComponent(saveAndRunButton, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE)
160: );
161:
162: jPanel1.add(jPanel2);
163:
164: saveAndExitButton.setText("Save and close");
165: saveAndExitButton.addActionListener(new java.awt.event.ActionListener() {
166:     public void actionPerformed(java.awt.event.ActionEvent evt) {
167:         saveAndExitButtonActionPerformed(evt);
168:     }
169: });
170:
171: javax.swing.GroupLayout jPanel3Layout = new javax.swing.GroupLayout(jPanel3);
172: jPanel3.setLayout(jPanel3Layout);
173: jPanel3Layout.setHorizontalGroup(
174:     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
175:         .addComponent(saveAndExitButton, javax.swing.GroupLayout.DEFAULT_SIZE, 222, Short.MAX_VALUE)
176: );
177: jPanel3Layout.setVerticalGroup(
178:     jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
179:         .addComponent(saveAndExitButton, javax.swing.GroupLayout.Alignment.TRAILING,
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE)
180:         );
181:
182:         jPanel1.add(jPanel3);
183:
184:         configurationList.setModel(new ConfigurationListModel(bIROAdaptorConfigurationList));
185:         configurationList.setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
186:         jScrollPane1.setViewportView(configurationList);
187:
188:         editButton.setText("Edit");
189:         editButton.addActionListener(new java.awt.event.ActionListener() {
190:             public void actionPerformed(java.awt.event.ActionEvent evt) {
191:                 editButtonActionPerformed(evt);
192:             }
193:         });
194:
195:         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
196:         this.setLayout(layout);
197:         layout.setHorizontalGroup(
198:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
199:                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
200:                     .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 350, Short.MAX_VALUE)
201:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
202:                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
203:                         .addComponent(newButton, javax.swing.GroupLayout.DEFAULT_SIZE, 63, Short.MAX_VALUE)
204:                         .addComponent(editButton, javax.swing.GroupLayout.DEFAULT_SIZE, 83, Short.MAX_VALUE)
205:                         .addComponent(copyButton, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
206:                     .addComponent(deleteButton, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
207:                 .addContainerGap()
208:                 .addGroup(layout.createSequentialGroup()
209:                     .addComponent(jLabel1)
210:                     .addContainerGap()
211:                     .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, 449, Short.MAX_VALUE)
212:                 );
213:         layout.setVerticalGroup(
214:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
215:                 .addGroup(layout.createSequentialGroup()
216:                     .addComponent(jLabel1)
217:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
218:                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
219:                         .addGroup(layout.createSequentialGroup()
220:                             .addComponent(newButton)
221:                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
222:         .addComponent(deleteButton)
223:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
224:         .addComponent(copyButton)
225:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
226:         .addComponent(editButton))
227:         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
228:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 151, Short.MAX_VALUE)
229:         .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE))
230:     };
231: } // </editor-fold> // GEN-END: initComponents
232: private void newButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_newButtonActionPerformed
233:     int count = 1;
234:     boolean nameAlreadyInUse = true;
235:     String proposedName = "";
236:     while (nameAlreadyInUse) {
237:         proposedName = "configuration" + count;
238:         nameAlreadyInUse = bIROAdaptorConfigurationList.containsConfigurationName(proposedName);
239:         count++;
240:     }
241:     JTextField configurationName = new JTextField(proposedName);
242:     int i = JOptionPane.showConfirmDialog(rootPanel, new Object[]{"Configuration name:", configurationName},
"New Adaptor configuration", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null);
243:     if (i == 0) {
244:         if (!bIROAdaptorConfigurationList.containsConfigurationName(configurationName.getText()) &&
!configurationName.getText().equals("")) {
245:             BIROAdaptorConfiguration conf = new BIROAdaptorConfiguration();
246:             conf.setName(configurationName.getText());
247:             ((ConfigurationListModel) configurationList.getModel()).addElement(conf);
248:         } else {
249:             JOptionPane.showMessageDialog(null, "This is not a valid configuration name", "Warning",
JOptionPane.WARNING_MESSAGE);
250:         }
251:     }
252: } // GEN-LAST: event_newButtonActionPerformed
253:
254: private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_deleteButtonActionPerformed
255:
256:     int i = JOptionPane.showConfirmDialog(rootPanel, new Object[]{"Do you really want to delete the selected
configuration?"}, "Delete configuration", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null);
257:     if (i == 0) {
258:         BIROAdaptorConfiguration conf = (BIROAdaptorConfiguration) configurationList.getSelectedValue();
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
259:            ((ConfigurationListModel) configurationList.getModel()).removeElement(conf);
260:        }
261:
262:    } //GEN-LAST:event_deleteButtonActionPerformed
263:
264:    private void copyButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_copyButtonActionPerformed
265:        int count = 2;
266:        boolean nameAlreadyInUse = true;
267:        String proposedName = "copy_of_" + ((BIROAdaptorConfiguration)
configurationList.getSelectedValue()).getName();
268:        nameAlreadyInUse = bIROAdaptorConfigurationList.containsConfigurationName(proposedName);
269:        while (nameAlreadyInUse) {
270:            proposedName = "copy_" + count + "_of_" + ((BIROAdaptorConfiguration)
configurationList.getSelectedValue()).getName();
271:            ;
272:            nameAlreadyInUse = bIROAdaptorConfigurationList.containsConfigurationName(proposedName);
273:            count++;
274:        }
275:        JTextField configurationName = new JTextField(proposedName);
276:        int i = JOptionPane.showConfirmDialog(rootPanel, new Object[]{"Create new configuration from selected one"
, "New configuration name:", configurationName}, "Copy configuration", JOptionPane.OK_CANCEL_OPTION,
JOptionPane.QUESTION_MESSAGE, null);
277:        if (i == 0) {
278:            if (!bIROAdaptorConfigurationList.containsConfigurationName(configurationName.getText()) &&
!configurationName.getText().equals("")) {
279:                BIROAdaptorConfiguration conf = (BIROAdaptorConfiguration) configurationList.getSelectedValue();
280:                BIROAdaptorConfiguration conf2 = (BIROAdaptorConfiguration) conf.clone();
281:                conf2.setName(configurationName.getText());
282:                ((ConfigurationListModel) configurationList.getModel()).addElement(conf2);
283:            } else {
284:                JOptionPane.showMessageDialog(null, "This is not a valid configuration name", "Warning",
JOptionPane.WARNING_MESSAGE);
285:            }
286:        }
287:    } //GEN-LAST:event_copyButtonActionPerformed
288:
289:    private void saveAndExitButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_saveAndExitButtonActionPerformed
290:        try {
291:            saveData();
292:
293:            bIROBoxConfiguration.saveToFile(new File(Configuration.bIROBoxConfigurationFilePath));
294:            bIRODatabaseManagerConfiguration.saveToFile(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
295:         bIROAdaptorConfigurationList.saveToFile(new File(Configuration.bIROAdaptorConfigurationListFilePath));
296:         JOptionPane.showMessageDialog(ConfigurationManagerPanel.this, "The configuration has been saved
correctly.", "Information", JOptionPane.INFORMATION_MESSAGE);
297:         rootPanel.addPanel(new HomePanel(rootPanel));
298:     } catch (Exception ex) {
299:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
300:         JOptionPane.showMessageDialog(ConfigurationManagerPanel.this, "The configuration has not been saved",
"Error", JOptionPane.ERROR_MESSAGE);
301:     }
302:
303: } //GEN-LAST:event_saveAndExitButtonActionPerformed
304:
305: private void editButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_editButtonActionPerformed
306:     try {
307:         saveData();
308:         rootPanel.addPanel(new ConnectionConfigurationPanel(rootPanel));
309:     } catch (Exception ex) {
310:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
311:     }
312: }
313:
314:
315: } //GEN-LAST:event_editButtonActionPerformed
316:
317: private void saveAndRunButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_saveAndRunButtonActionPerformed
318:     try {
319:
320:         saveData();
321:         bIROAdaptorConfigurationList.saveToFile(new File(Configuration.bIROAdaptorConfigurationListFilePath));
322:         bIROBoxConfiguration.saveToFile(new File(Configuration.bIROBoxConfigurationFilePath));
323:         bIRODatabaseManagerConfiguration.saveToFile(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
324:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
325:         new Thread(new Runnable() {
326:
327:             public void run() {
328:
329:                 BIROAdaptor2 ad = new BIROAdaptor2();
330:                 AdaptorProgressPanel panel = AdaptorProgressPanel.createAndShowAdaptorProgress(ad);
331:                 try {
332:                     panel.showText("Adaptor is reading the configuration named: " +
(bIROAdaptorConfigurationList.getBIROAdaptorConfigurationList().getCurrentConfiguration().getName());
333:                     BIROAdaptorConfiguration bIROAdaptorConfiguration =
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java

```
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration();
334:         checkPrimaryKey();
335:         ad.connectToDatabase(bIROAdaptorConfiguration.getDBMSDriver(),
bIROAdaptorConfiguration.getMergeTableQuery(), bIROAdaptorConfiguration.getActivityTableQuery());
336:         ad.initFileOut(bIROAdaptorConfiguration.getExportFilePath());
337:         ad.exportBIROFields(bIROAdaptorConfiguration.getBIROFieldList());
338:         panel.showText("Adaptor process completed successfully");
339:
340:     } catch (SQLException ex) {
341:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
342:         panel.showText("Something wrong with SQL Access for:\n" + ex.toString());
343:     } catch (IOException ex) {
344:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
345:         panel.showText("Something wrong with IO Access for:\n" + ex.toString());
346:     } catch (ClassNotFoundException ex) {
347:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
348:         panel.showText("Something wrong with JDBC Driver for:\n" + ex.toString());
349:     } catch (Exception ex) {
350:         Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
351:         panel.showText("Something went wrong: " + ex.getMessage());
352:     } finally {
353:         setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
354:     }
355: }
356: }).start();
357:
358: } catch (Exception ex) {
359:     JOptionPane.showMessageDialog(this, "An error occurred when saving the configuration.\nPlease try
again.", "Error", JOptionPane.ERROR_MESSAGE);
360:     Logger.getLogger(ConfigurationManagerPanel.class.getName()).log(Level.SEVERE, null,
ex); //GEN-LAST:event_saveAndRunButtonActionPerformed
361: }
362: }
363:
364: private void checkPrimaryKey() throws Exception{
365:     boolean constraintViolated= false;
366:     String duplicates= "";
367:     BIROAdaptorConfiguration bIROAdaptorConfiguration =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration();
368:     ConnectionManager manager = ConnectionManager.initializeManager(bIROAdaptorConfiguration.getDBMSDriver());
369:
370:     String idColumnName = ((BIROField)
bIROAdaptorConfiguration.getBIROFieldList().getPatientID()).getDBColumnName();
371:     String episodeDateColumnName = ((BIROField)
bIROAdaptorConfiguration.getBIROFieldList().getEpisodeDate()).getDBColumnName();
```

```
372:         String mergeTableQuery = bIROAdaptorConfiguration.getMergeTableQuery();
373:         String query = "SELECT \" + idColumnName + "\",\" + episodeDateColumnName + "\" FROM (\" +
mergeTableQuery + ") AS merge_table_query GROUP BY \" + idColumnName + "\",\" + episodeDateColumnName + "\" HAVING
COUNT(\" + episodeDateColumnName + "\">1";
374:         //System.out.println(query);
375:         ResultSet set = manager.executeQuery(query);
376:         manager.close();
377:         while (set.next())
378:         {
379:             constraintViolated = true;
380:             duplicates+=(set.getString(idColumnName)+", "+set.getString(episodeDateColumnName)+"\n") ;
381:         }
382:         if(constraintViolated){
383:             throw new Exception("Duplicated values for patient ID and Episode Date found.\nPlease check the
following values:\n"+duplicates);
384:         }
385:
386:     }
387:     // Variables declaration - do not modify//GEN-BEGIN:variables
388:     private javax.swing.JList configurationList;
389:     private javax.swing.JButton copyButton;
390:     private javax.swing.JButton deleteButton;
391:     private javax.swing.JButton editButton;
392:     private javax.swing.JLabel jLabel1;
393:     private javax.swing.JPanel jPanel1;
394:     private javax.swing.JPanel jPanel2;
395:     private javax.swing.JPanel jPanel3;
396:     private javax.swing.JScrollPane jScrollPane1;
397:     private javax.swing.JButton newButton;
398:     private javax.swing.JButton saveAndExitButton;
399:     private javax.swing.JButton saveAndRunButton;
400:     // End of variables declaration//GEN-END:variables
401:
402: private void loadData() throws IOException {
403:     BIROAdaptorConfiguration b = bIROAdaptorConfigurationList.getCurrentConfiguration();
404:     if (b != null) {
405:         configurationList.setSelectedValue(b, true);
406:     }
407:     if (configurationList.getSelectedIndex() == -1) {
408:         //No selection, disable fire button.
409:         copyButton.setEnabled(false);
410:         editButton.setEnabled(false);
411:         deleteButton.setEnabled(false);
412:         saveAndRunButton.setEnabled(false);
413:     } else {
```

```
414:            //Selection, enable the fire button.
415:            copyButton.setEnabled(true);
416:            editButton.setEnabled(true);
417:            deleteButton.setEnabled(true);
418:            saveAndRunButton.setEnabled(true);
419:        }
420:    }
421:
422:    @Override
423:    public void saveData() throws Exception {
424:        bIROAdaptorConfigurationList.setCurrentConfiguration((BIROAdaptorConfiguration)
configurationList.getSelectedValue());
425:    }
426: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ConnectionConfigurationPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.birobox.utils.ConnectionManager;
34: import eu.biro.birobox.configuration.BIROBoxConfiguration;
35: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
36: import eu.biro.birobox.utils.CustomDBMSDriver;
37: import eu.biro.birobox.panel.*;
38: import eu.biro.birobox.main.*;
39: import eu.biro.birobox.panel.ChildrenPanel;
40: import eu.biro.adaptor2.DBMSDriver;
41: import eu.biro.adaptor2.BIROAdaptorConfiguration;
42: import eu.biro.adaptor2.CSVFILEDriverListener;
43: import eu.biro.adaptor2.Drivers.CSVFILEDriver;
44: import eu.biro.birobox.models.DBMSDriverComboBoxModel;
45: import java.awt.BorderLayout;
```

```
46: import java.awt.Cursor;
47: import java.io.File;
48: import java.io.IOException;
49: import java.sql.SQLException;
50: import java.util.Vector;
51: import java.util.logging.Level;
52: import java.util.logging.Logger;
53: import javax.swing.JButton;
54: import javax.swing.JFileChooser;
55: import javax.swing.JFrame;
56: import javax.swing.JOptionPane;
57: import javax.swing.JScrollPane;
58: import javax.swing.JTextArea;
59: import javax.swing.JTextField;
60: import javax.swing.SwingUtilities;
61: import javax.swing.filechooser.FileFilter;
62: import javax.swing.filechooser.FileNameExtensionFilter;
63:
64:
65: public class ConnectionConfigurationPanel extends ChildrenPanel implements CSVFILEDriverListener {
66:
67:     private RootPanel rootPanel;
68:     private BIROAdaptorConfiguration bIROAdaptorConfiguration;
69:     private BIROAdaptorConfigurationList bIROAdaptorConfigurationList;
70:     private BIROBoxConfiguration bIROBoxConfiguration;
71:     private BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
72:     private Vector<DBMSDriver> dbMSDriverVector;
73:
74:     /** Creates new form NewJPanel */
75:     public ConnectionConfigurationPanel(RootPanel rootPanel) throws IOException {
76:         this.rootPanel = rootPanel;
77:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
78:         bIROAdaptorConfigurationList = BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList();
79:         bIROAdaptorConfiguration = bIROAdaptorConfigurationList.getCurrentConfiguration();
80:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
81:         dbMSDriverVector = bIROBoxConfiguration.getDBMSDriverVector();
82:         initComponents();
83:         loadData();
84:     }
85:
86:     /** This method is called from within the constructor to
87:      * initialize the form.
88:      * WARNING: Do NOT modify this code. The content of this method is
89:      * always regenerated by the Form Editor.
90:      */
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConnectionConfigurationPanel.java

```
91: // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
92: private void initComponents() {
93:     bindingGroup = new org.jdesktop.beansbinding.BindingGroup();
94:
95:     buttonGroup1 = new javax.swing.ButtonGroup();
96:     jdbcConfigurationPanel = new javax.swing.JPanel();
97:     jdbcDriverLabel = new javax.swing.JLabel();
98:     jdbcDriverComboBox = new javax.swing.JComboBox();
99:     databaseHostAndPortLabel = new javax.swing.JLabel();
100:    databaseHostAndPortTextField = new javax.swing.JTextField();
101:    databaseUsernameLabel = new javax.swing.JLabel();
102:    databaseUsernameTextField = new javax.swing.JTextField();
103:    databasePasswordLabel = new javax.swing.JLabel();
104:    databasePasswordField = new javax.swing.JPasswordField();
105:    descriptionLabel = new javax.swing.JLabel();
106:    buttonPanel = new javax.swing.JPanel();
107:    leftButtonPanel = new javax.swing.JPanel();
108:    previousButton = new javax.swing.JButton();
109:    rightButtonPanel = new javax.swing.JPanel();
110:    nextButton = new javax.swing.JButton();
111:    databaseNameTextField = new javax.swing.JTextField();
112:    databaseNameLabel = new javax.swing.JLabel();
113:    addDriverButton = new javax.swing.JButton();
114:    removeDriverButton = new javax.swing.JButton();
115:    mergeFileLabel = new javax.swing.JLabel();
116:    mergeFileTextField = new javax.swing.JTextField();
117:    mergeFileBrowseButton = new javax.swing.JButton();
118:    databaseRadioButton = new javax.swing.JRadioButton();
119:    csvfileRadioButton = new javax.swing.JRadioButton();
120:    separatorComboBox = new javax.swing.JComboBox();
121:    separatorLabel = new javax.swing.JLabel();
122:    activityFileTextField = new javax.swing.JTextField();
123:    activityFileBrowseButton = new javax.swing.JButton();
124:    activityFileLabel = new javax.swing.JLabel();
125:    jLabel1 = new javax.swing.JLabel();
126:
127:    jdbcConfigurationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Connection configuration"));
128:
129:    jdbcDriverLabel.setText("DBMS driver *");
130:
131:    org.jdesktop.beansbinding.Binding binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE,
jdbcDriverComboBox, org.jdesktop.beansbinding.ObjectProperty.create(), jdbcDriverLabel,
org.jdesktop.beansbinding.BeanProperty.create("labelFor"));
132:    bindingGroup.addBinding(binding);
```

```
133:
134:         jdbcDriverComboBox.setModel(new DBMSDriverComboBoxModel(dbMSDriverVector));
135:
136:         databaseHostAndPortLabel.setText("database host and port *");
137:
138:         binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE,
databaseHostAndPortTextField, org.jdesktop.beansbinding.ObjectProperty.create(), databaseHostAndPortLabel,
org.jdesktop.beansbinding.BeanProperty.create("labelFor"));
139:         bindingGroup.addBinding(binding);
140:
141:         databaseUsernameLabel.setText("database username *");
142:
143:         binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE,
databaseUsernameTextField, org.jdesktop.beansbinding.ObjectProperty.create(), databaseUsernameLabel,
org.jdesktop.beansbinding.BeanProperty.create("labelFor"));
144:         bindingGroup.addBinding(binding);
145:
146:         databasePasswordLabel.setText("database password *");
147:
148:         descriptionLabel.setText("Configure the connection to data source");
149:
150:         buttonPanel.setLayout(new java.awt.GridLayout(1, 2, 5, 5));
151:
152:         leftButtonPanel.setPreferredSize(new java.awt.Dimension(23, 0));
153:
154:         previousButton.setText("Previous");
155:         previousButton.addActionListener(new java.awt.event.ActionListener() {
156:             public void actionPerformed(java.awt.event.ActionEvent evt) {
157:                 previousButtonActionPerformed(evt);
158:             }
159:         });
160:
161:         javax.swing.GroupLayout leftButtonPanelLayout = new javax.swing.GroupLayout(leftButtonPanel);
162:         leftButtonPanel.setLayout(leftButtonPanelLayout);
163:         leftButtonPanelLayout.setHorizontalGroup(
164:             leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
165:                 .addComponent(previousButton, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 258, Short.MAX_VALUE)
166:         );
167:         leftButtonPanelLayout.setVerticalGroup(
168:             leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
169:                 .addComponent(previousButton, javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE)
170:         );
```

```
171:
172:     buttonPanel.add(leftButtonPanel);
173:
174:     nextButton.setText("Next");
175:     nextButton.addActionListener(new java.awt.event.ActionListener() {
176:         public void actionPerformed(java.awt.event.ActionEvent evt) {
177:             nextButtonActionPerformed(evt);
178:         }
179:     });
180:
181:     javax.swing.GroupLayout rightButtonPanelLayout = new javax.swing.GroupLayout(rightButtonPanel);
182:     rightButtonPanel.setLayout(rightButtonPanelLayout);
183:     rightButtonPanelLayout.setHorizontalGroup(
184:         rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
185:             .addComponent(nextButton, javax.swing.GroupLayout.DEFAULT_SIZE, 258, Short.MAX_VALUE)
186:     );
187:     rightButtonPanelLayout.setVerticalGroup(
188:         rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
189:             .addComponent(nextButton, javax.swing.GroupLayout.DEFAULT_SIZE, 26, Short.MAX_VALUE)
190:     );
191:
192:     buttonPanel.add(rightButtonPanel);
193:
194:     databaseNameLabel.setText("database name *");
195:
196:     binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE,
databaseNameTextField, org.jdesktop.beansbinding.ObjectProperty.create(), databaseNameLabel,
org.jdesktop.beansbinding.BeanProperty.create("labelFor"));
197:     bindingGroup.addBinding(binding);
198:
199:     addDriverButton.setText("+");
200:     addDriverButton.addActionListener(new java.awt.event.ActionListener() {
201:         public void actionPerformed(java.awt.event.ActionEvent evt) {
202:             addDriverButtonActionPerformed(evt);
203:         }
204:     });
205:
206:     removeDriverButton.setText("-");
207:     removeDriverButton.addActionListener(new java.awt.event.ActionListener() {
208:         public void actionPerformed(java.awt.event.ActionEvent evt) {
209:             removeDriverButtonActionPerformed(evt);
210:         }
211:     });
212:
```

```
213: mergeFileLabel.setText("merge file *");
214:
215: mergeFileBrowseButton.setText("Browse");
216: mergeFileBrowseButton.addActionListener(new java.awt.event.ActionListener() {
217:     public void actionPerformed(java.awt.event.ActionEvent evt) {
218:         mergeFileBrowseButtonActionPerformed(evt);
219:     }
220: });
221:
222: buttonGroup1.add(databaseRadioButton);
223: databaseRadioButton.setText("database");
224: databaseRadioButton.addActionListener(new java.awt.event.ActionListener() {
225:     public void actionPerformed(java.awt.event.ActionEvent evt) {
226:         databaseRadioButtonActionPerformed(evt);
227:     }
228: });
229:
230: buttonGroup1.add(csvfileRadioButton);
231: csvfileRadioButton.setText("csv file");
232: csvfileRadioButton.addActionListener(new java.awt.event.ActionListener() {
233:     public void actionPerformed(java.awt.event.ActionEvent evt) {
234:         csvfileRadioButtonActionPerformed(evt);
235:     }
236: });
237:
238: separatorComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { ",", ";", "|" }));
239:
240: separatorLabel.setText("separator*");
241:
242: activityFileBrowseButton.setText("Browse");
243: activityFileBrowseButton.addActionListener(new java.awt.event.ActionListener() {
244:     public void actionPerformed(java.awt.event.ActionEvent evt) {
245:         activityFileBrowseButtonActionPerformed(evt);
246:     }
247: });
248:
249: activityFileLabel.setText("activity file name ");
250:
251: jLabel1.setText("*= required fields");
252:
253: javax.swing.GroupLayout jdbcConfigurationPanelLayout = new javax.swing.GroupLayout(jdbcConfigurationPanel);
254: jdbcConfigurationPanel.setLayout(jdbcConfigurationPanelLayout);
255: jdbcConfigurationPanelLayout.setHorizontalGroup(
256:     jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
257:         .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup()
```

```
258:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
259:         .addComponent(descriptionLabel)
260:         .addComponent(databaseRadioButton)
261:         .addComponent(csvfileRadioButton)
262:         .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 522, Short.MAX_VALUE)
263:         .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup())
264:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
265:         .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
266:         .addComponent(databasePasswordLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
267:         .addComponent(databaseHostAndPortLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 135, Short.MAX_VALUE)
268:         .addComponent(jdbcDriverLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
269:         .addComponent(databaseNameLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
270:         .addComponent(separatorLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
271:         .addComponent(activityFileLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
272:         .addComponent(mergeFileLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
273:         .addComponent(databaseUsernameLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
274:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
275:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
276:         .addComponent(separatorComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
277:         .addComponent(databaseNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 383,
Short.MAX_VALUE)
278:         .addComponent(databaseUsernameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 383,
Short.MAX_VALUE)
279:         .addComponent(databasePasswordField, javax.swing.GroupLayout.DEFAULT_SIZE, 383,
Short.MAX_VALUE)
280:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jdbcConfigurationPanelLayout.createSequentialGroup())
281:         .addComponent(jdbcDriverComboBox, 0, 286, Short.MAX_VALUE)
282:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
283:         .addComponent(addDriverButton, javax.swing.GroupLayout.PREFERRED_SIZE, 43,
javax.swing.GroupLayout.PREFERRED_SIZE)
284:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
285:                                .addComponent(removeDriverButton, javax.swing.GroupLayout.PREFERRED_SIZE, 42,
javax.swing.GroupLayout.PREFERRED_SIZE))
286:                                .addComponent(databaseHostAndPortTextField,
javax.swing.GroupLayout.Alignment.TRAILING, javax.swing.GroupLayout.DEFAULT_SIZE, 383, Short.MAX_VALUE)
287:                                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jdbcConfigurationPanelLayout.createSequentialGroup())
288:                                .addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
289:                                .addComponent(activityFileTextField,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 310, Short.MAX_VALUE)
290:                                .addComponent(mergeFileTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 310,
Short.MAX_VALUE))
291:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
292:                                .addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
293:                                .addComponent(activityFileBrowseButton, 0,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
294:                                .addComponent(mergeFileBrowseButton, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))))
295:                                .addContainerGap())
296:                                );
297:                                jdbcConfigurationPanelLayout.setVerticalGroup(
298:                                jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
299:                                .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup())
300:                                .addComponent(descriptionLabel)
301:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
302:                                .addComponent(databaseRadioButton)
303:                                .addGap(9, 9, 9)
304:                                .addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
305:                                .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup())
306:                                .addComponent(jdbcDriverLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
307:                                .addGap(10, 10, 10)
308:                                .addComponent(databaseHostAndPortLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
309:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
310:                                .addComponent(databaseNameLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
311:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
312:                                .addComponent(databaseUsernameLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
313:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
314:                                .addComponent(databasePasswordLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

```
315:                .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup())
316:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
317:                .addComponent(jdbcDriverComboBox, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
318:                .addComponent(addDriverButton)
319:                .addComponent(removeDriverButton))
320:                .addGap(10, 10, 10)
321:                .addComponent(databaseHostAndPortTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
322:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
323:                .addComponent(databaseNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
324:                .addGap(6, 6, 6)
325:                .addComponent(databaseUsernameTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
326:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
327:                .addComponent(databasePasswordField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)))
328:                .addGap(17, 17, 17)
329:                .addComponent(csvfileRadioButton)
330:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
331:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
332:                .addComponent(mergeFileBrowseButton)
333:                .addComponent(mergeFileTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
334:                .addComponent(mergeFileLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
335:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
336:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
337:                .addComponent(activityFileTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
338:                .addComponent(activityFileBrowseButton)
339:                .addComponent(activityFileLabel))
340:                .addGap(12, 12, 12)
341:
.addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
342:                .addComponent(separatorComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
343:                .addComponent(separatorLabel))
344:                .addGap(18, 18, 18)
345:                .addComponent(jLabel1)
346:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 96, Short.MAX_VALUE)
```

```
347:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
348:     );
349:
350:     jdbcConfigurationPanelLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{activityFileTextField, databaseHostAndPortTextField, databaseNameTextField, databasePasswordField,
databaseUsernameTextField, jdbcDriverComboBox, mergeFileTextField});
351:
352:     jdbcConfigurationPanelLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{databaseHostAndPortLabel, databaseNameLabel, databasePasswordLabel, databaseUsernameLabel, jdbcDriverLabel,
mergeFileLabel});
353:
354:     jdbcConfigurationPanelLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{activityFileBrowseButton, mergeFileBrowseButton});
355:
356:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
357:     this.setLayout(layout);
358:     layout.setHorizontalGroup(
359:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
360:         .addComponent(jdbcConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
361:     );
362:     layout.setVerticalGroup(
363:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
364:         .addComponent(jdbcConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 511, Short.MAX_VALUE)
365:     );
366:
367:     bindingGroup.bind();
368: } // </editor-fold> // GEN-END: initComponents
369: private void nextButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_nextButtonActionPerformed
370:     try {
371:         saveData();
372:         checkConnection();
373:         rootPanel.addPanel(new MergeTableConfigurationPanel(rootPanel));
374:     } catch (Exception ex) {
375:         Logger.getLogger(ConnectionConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
376:         JOptionPane.showMessageDialog(rootPanel, "I cannot establish a connection to data source.\nPlease
check the configuration.\n" + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE, null);
377:     }
378: } // GEN-LAST: event_nextButtonActionPerformed
379:
380: private void addDriverButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_addDriverButtonActionPerformed
381:     JTextField dBMSNameTextField = new JTextField();
```

BIROBox/src/eu/biro/birobox/panel/adaptor/ConnectionConfigurationPanel.java

```
382:         JTextField driverClassTextField = new JTextField();
383:         JTextField driverUrlPatternTextField = new JTextField();
384:         JTextField driverJarPathTextField = new JTextField();
385:         //JButton browseJarButton = new JButton("browse");
386:         int i = JOptionPane.showConfirmDialog(rootPanel, new Object[]{"DBMS name:", dBMSNameTextField, "driver
class:", driverClassTextField, "driver url pattern:", driverUrlPatternTextField, "jar path:", driverJarPathTextField
/*,browseJarButton*/}, "New DBMS Driver", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null);
387:         if (i == 0) {
388:             if (!dBMSNameTextField.getText().equals("") && !driverClassTextField.getText().equals("") &&
!driverUrlPatternTextField.getText().equals("") && !driverJarPathTextField.getText().equals("")) {
389:                 ((DBMSDriverComboBoxModel) jdbcDriverComboBox.getModel()).addElement(new
CustomDBMSDriver(dBMSNameTextField.getText(), driverClassTextField.getText(), driverUrlPatternTextField.getText(),
driverJarPathTextField.getText()));
390:                 jdbcDriverComboBox.setSelectedIndex(((DBMSDriverComboBoxModel)
(jdbcDriverComboBox.getModel()).getSize() - 1);
391:                 jdbcDriverComboBox.repaint();
392:             }
393:         }
394:     }
395:
396: } //GEN-LAST:event_addDriverButtonActionPerformed
397:
398: private void removeDriverButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_removeDriverButtonActionPerformed
399:     DBMSDriver d = (DBMSDriver) jdbcDriverComboBox.getSelectedItem();
400:     boolean stillInUse = false;
401:
402:     if (d.compare(bIRODatabaseManagerConfiguration.getBIRODBMSDriver()) ||
(bIROAdaptorConfigurationList.getDBMSDriverUsersNumber(d) > 1)) {
403:         stillInUse = true;
404:     }
405:     if (!stillInUse) {
406:         int i = JOptionPane.showConfirmDialog(rootPanel, "Do you really want to remove/n the selected driver?"
, "Remove", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null);
407:         if (i == 0) {
408:             int selectedIndex = jdbcDriverComboBox.getSelectedIndex();
409:             ((DBMSDriverComboBoxModel) (jdbcDriverComboBox.getModel())).removeElementAt(selectedIndex);
410:             if (selectedIndex != 0) {
411:                 jdbcDriverComboBox.setSelectedIndex(selectedIndex - 1);
412:             }
413:             jdbcDriverComboBox.repaint();
414:         }
415:     } else {
416:         JOptionPane.showMessageDialog(rootPanel, "This driver cannot be removed because is still in use",
"Warning", JOptionPane.WARNING_MESSAGE);
```

```
417:         }
418:     } //GEN-LAST:event_removeDriverButtonActionPerformed
419:
420:     private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_previousButtonActionPerformed
421:
422:         try {
423:             saveData();
424:             rootPanel.addPanel(new ConfigurationManagerPanel(rootPanel));
425:         } catch (Exception ex) {
426:             Logger.getLogger(ConnectionConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
427:             JOptionPane.showMessageDialog(rootPanel, "I cannot establish a connection.\nPlease check JDBC
Configuration.\n Error: " + ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE, null);
428:         }
429:
430:
431:     } //GEN-LAST:event_previousButtonActionPerformed
432:
433:     private void databaseRadioButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_databaseRadioButtonActionPerformed
434:         mergeFileBrowseButton.setEnabled(false);
435:         mergeFileTextField.setEnabled(false);
436:         mergeFileLabel.setEnabled(false);
437:         activityFileBrowseButton.setEnabled(false);
438:         activityFileTextField.setEnabled(false);
439:         activityFileLabel.setEnabled(false);
440:         separatorComboBox.setEnabled(false);
441:         separatorLabel.setEnabled(false);
442:
443:         jdbcDriverComboBox.setEnabled(true);
444:         jdbcDriverLabel.setEnabled(true);
445:         databaseHostAndPortTextField.setEditable(true);
446:         databaseHostAndPortLabel.setEnabled(true);
447:         databaseNameTextField.setEnabled(true);
448:         databaseNameLabel.setEnabled(true);
449:         databaseUsernameTextField.setEnabled(true);
450:         databaseUsernameLabel.setEnabled(true);
451:         databasePasswordField.setEnabled(true);
452:         databasePasswordLabel.setEnabled(true);
453:         addDriverButton.setEnabled(true);
454:         removeDriverButton.setEnabled(true);
455:
456:     } //GEN-LAST:event_databaseRadioButtonActionPerformed
457:
458:     private void csvfileRadioButtonActionPerformed(java.awt.event.ActionEvent evt)
```

```
{//GEN-FIRST:event_csvfileRadioButtonActionPerformed
459:         mergeFileBrowseButton.setEnabled(true);
460:         mergeFileTextField.setEnabled(true);
461:         mergeFileLabel.setEnabled(true);
462:         activityFileBrowseButton.setEnabled(true);
463:         activityFileTextField.setEnabled(true);
464:         activityFileLabel.setEnabled(true);
465:         separatorComboBox.setEnabled(true);
466:         separatorLabel.setEnabled(true);
467:
468:         jdbcDriverComboBox.setEnabled(false);
469:         jdbcDriverLabel.setEnabled(false);
470:         databaseHostAndPortTextField.setEditable(false);
471:         databaseHostAndPortLabel.setEnabled(false);
472:         databaseNameTextField.setEnabled(false);
473:         databaseNameLabel.setEnabled(false);
474:         databaseUsernameTextField.setEnabled(false);
475:         databaseUsernameLabel.setEnabled(false);
476:         databasePasswordField.setEnabled(false);
477:         databasePasswordLabel.setEnabled(false);
478:         addDriverButton.setEnabled(false);
479:         removeDriverButton.setEnabled(false);
480:
481:
482:     }//GEN-LAST:event_csvfileRadioButtonActionPerformed
483:
484:     private void mergeFileBrowseButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_mergeFileBrowseButtonActionPerformed
485:
486:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
487:         SwingUtilities.invokeLater(new Runnable() {
488:
489:             public void run() {
490:                 JFileChooser fc = new JFileChooser();
491:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("CSV file", "csv"));
492:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
493:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
494:                 fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
495:                 int returnVal = fc.showOpenDialog(ConnectionConfigurationPanel.this);
496:                 if (returnVal == JFileChooser.APPROVE_OPTION) {
497:                     File file = fc.getSelectedFile();
498:                     mergeFileTextField.setText(file.getAbsolutePath());
499:                 }
500:                 setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
501:             }
```

```
502:         });
503:
504: } //GEN-LAST:event_mergeFileBrowseButtonActionPerformed
505:
506:     private void activityFileBrowseButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_activityFileBrowseButtonActionPerformed
507:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
508:         SwingUtilities.invokeLater(new Runnable() {
509:
510:             public void run() {
511:                 JFileChooser fc = new JFileChooser();
512:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("CSV file", "csv"));
513:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
514:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
515:                 fc.setSelectionMode(JFileChooser.FILES_ONLY);
516:                 int returnVal = fc.showOpenDialog(ConnectionConfigurationPanel.this);
517:                 if (returnVal == JFileChooser.APPROVE_OPTION) {
518:                     File file = fc.getSelectedFile();
519:                     activityFileTextField.setText(file.getAbsolutePath());
520:                 }
521:                 setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
522:             }
523:         });
524:
525: } //GEN-LAST:event_activityFileBrowseButtonActionPerformed
526:
527:     private void loadData() throws IOException {
528:         DBMSDriver d = bIROAdaptorConfiguration.getDBMSDriver();
529:         if (d == null) {
530:             databaseRadioButton.setSelected(true);
531:             csvfileRadioButton.setSelected(false);
532:             jdbcDriverComboBox.setSelectedIndex(0);
533:             mergeFileBrowseButton.setEnabled(false);
534:             mergeFileTextField.setEnabled(false);
535:             mergeFileLabel.setEnabled(false);
536:             activityFileBrowseButton.setEnabled(false);
537:             activityFileTextField.setEnabled(false);
538:             activityFileLabel.setEnabled(false);
539:             separatorComboBox.setEnabled(false);
540:             separatorLabel.setEnabled(false);
541:         } else if (d != null && !(d instanceof CSVFILEDriver)) {
542:             databaseRadioButton.setSelected(true);
543:             ((DBMSDriverComboBoxModel) jdbcDriverComboBox.getModel()).setSelectedDriver(d);
544:             databaseHostAndPortTextField.setText(d.getHostAndPort());
545:             databaseNameTextField.setText(d.getDatabaseName());
```

```
546:         databaseUsernameTextField.setText(d.getUser());
547:         databasePasswordField.setText(d.getPwd());
548:         mergeFileBrowseButton.setEnabled(false);
549:         mergeFileTextField.setEnabled(false);
550:         mergeFileLabel.setEnabled(false);
551:         activityFileBrowseButton.setEnabled(false);
552:         activityFileTextField.setEnabled(false);
553:         activityFileLabel.setEnabled(false);
554:         separatorComboBox.setEnabled(false);
555:         separatorLabel.setEnabled(false);
556:     } else if (d != null && (d instanceof CSVFILEDriver)) {
557:         csvfileRadioButton.setSelected(true);
558:         mergeFileBrowseButton.setEnabled(true);
559:         activityFileBrowseButton.setEnabled(true);
560:         separatorComboBox.setEnabled(true);
561:
562:         jdbcDriverComboBox.setEnabled(false);
563:         jdbcDriverLabel.setEnabled(false);
564:         databaseHostAndPortTextField.setEditable(false);
565:         databaseHostAndPortLabel.setEnabled(false);
566:         databaseNameTextField.setEnabled(false);
567:         databaseNameLabel.setEnabled(false);
568:         databaseUsernameTextField.setEnabled(false);
569:         databaseUsernameLabel.setEnabled(false);
570:         databasePasswordField.setEnabled(false);
571:         databasePasswordLabel.setEnabled(false);
572:         addDriverButton.setEnabled(false);
573:         removeDriverButton.setEnabled(false);
574:         mergeFileTextField.setText(((CSVFILEDriver) d).getMergeFileName());
575:         activityFileTextField.setText(((CSVFILEDriver) d).getActivityFileName());
576:         separatorComboBox.setSelectedItem(((CSVFILEDriver) d).getSeparator());
577:
578:     }
579: }
580:
581: public void saveData() throws Exception {
582:     if (databaseRadioButton.isSelected()) {
583:         DBMSDriver driver = ((DBMSDriver) jdbcDriverComboBox.getModel().getSelectedItem());
584:         DBMSDriver driverClone = (DBMSDriver) driver.clone();
585:         driverClone.setHostAndPort(databaseHostAndPortTextField.getText());
586:         driverClone.setDatabaseName(databaseNameTextField.getText());
587:         driverClone.setUser(databaseUsernameTextField.getText());
588:         driverClone.setPwd(String.valueOf(databasePasswordField.getPassword()));
589:         bIROAdaptorConfiguration.setDBMSDriver(driverClone);
590:         bIROBoxConfiguration.setDBMSDriverVector(driverClone);
```

```
591:         } else if (csvfileRadioButton.isSelected()) {
592:             CSVFILEDriver driver = new CSVFILEDriver(mergeFileTextField.getText(),
activityFileTextField.getText(), (String) separatorComboBox.getSelectedItem());
593:             bIROAdaptorConfiguration.setDBMSDriver(driver);
594:             bIROBoxConfiguration.setDBMSDriverVector(dBMSDriverVector);
595:         }
596:
597:     }
598:
599:     private void checkConnection() throws ClassNotFoundException, SQLException, Exception {
600:         ConnectionManager manager = ConnectionManager.initializeManager(bIROAdaptorConfiguration.getDBMSDriver());
601:
602:         if (bIROAdaptorConfiguration.getDBMSDriver() instanceof CSVFILEDriver) {
603:             ((CSVFILEDriver) bIROAdaptorConfiguration.getDBMSDriver()).addProgressListener(this);
604:         }
605:         manager.connect();
606:         if (bIROAdaptorConfiguration.getDBMSDriver() instanceof CSVFILEDriver) {
607:             ((CSVFILEDriver) bIROAdaptorConfiguration.getDBMSDriver()).removeProgressListener(this);
608:         }
609:
610:     }
611:     // Variables declaration - do not modify//GEN-BEGIN:variables
612:     private javax.swing.JButton activityFileBrowseButton;
613:     private javax.swing.JLabel activityFileLabel;
614:     private javax.swing.JTextField activityFileTextField;
615:     private javax.swing.JButton addDriverButton;
616:     private javax.swing.ButtonGroup buttonGroup1;
617:     private javax.swing.JPanel buttonPanel;
618:     private javax.swing.JRadioButton csvfileRadioButton;
619:     private javax.swing.JLabel databaseHostAndPortLabel;
620:     private javax.swing.JTextField databaseHostAndPortTextField;
621:     private javax.swing.JLabel databaseNameLabel;
622:     private javax.swing.JTextField databaseNameTextField;
623:     private javax.swing.JPasswordField databasePasswordField;
624:     private javax.swing.JLabel databasePasswordLabel;
625:     private javax.swing.JRadioButton databaseRadioButton;
626:     private javax.swing.JLabel databaseUsernameLabel;
627:     private javax.swing.JTextField databaseUsernameTextField;
628:     private javax.swing.JLabel descriptionLabel;
629:     private javax.swing.JLabel jLabel1;
630:     private javax.swing.JPanel jdbcConfigurationPanel;
631:     private javax.swing.JComboBox jdbcDriverComboBox;
632:     private javax.swing.JLabel jdbcDriverLabel;
633:     private javax.swing.JPanel leftButtonPanel;
634:     private javax.swing.JButton mergeFileBrowseButton;
```

```
635:     private javax.swing.JLabel mergeFileLabel;
636:     private javax.swing.JTextField mergeFileTextField;
637:     private javax.swing.JButton nextButton;
638:     private javax.swing.JButton previousButton;
639:     private javax.swing.JButton removeDriverButton;
640:     private javax.swing.JPanel rightButtonPanel;
641:     private javax.swing.JComboBox separatorComboBox;
642:     private javax.swing.JLabel separatorLabel;
643:     private org.jdesktop.beansbinding.BindingGroup bindingGroup;
644:     // End of variables declaration//GEN-END:variables
645:     public void showText(String txt) {
646:
647:         JOptionPane.showMessageDialog(this,new JScrollPane(new JTextArea(txt,10, 50)), "WARNING",
JOptionPane.WARNING_MESSAGE);
648:
649:
650:
651:     }
652: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      DataSourceConfigurationPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.panel.*;
33: import eu.biro.birobox.panel.adaptor.FieldMappingPanel;
34:
35: import eu.biro.adaptor2.field.BIROField;
36: import eu.biro.adaptor2.field.BIROFieldList;
37: import eu.biro.adaptor2.field.BIROFieldType;
38: import eu.biro.adaptor2.field.StaticBIROField;
39: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
40: import java.util.logging.Level;
41: import java.util.logging.Logger;
42: import javax.swing.ListSelectionModel;
43: import eu.biro.birobox.models.StaticFieldTableCellRenderer;
44: import eu.biro.birobox.models.StaticFieldTableModel;
45: import java.util.Iterator;
```

```
46: import javax.swing.JOptionPane;
47:
48: public class DataSourceConfigurationPanel extends ChildrenPanel {
49:
50:     private RootPanel rootPanel;
51:     private BIROFieldList l;
52:
53:     /** Creates new form NewJPanel3 */
54:     public DataSourceConfigurationPanel(RootPanel rootPanel) {
55:         this.rootPanel = rootPanel;
56:         initComponents();
57:         ListSelectionModel listSelectionModel;
58:         listSelectionModel = siteHeaderTable.getSelectionModel();
59:         listSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
60:         listSelectionModel = siteProfileTable.getSelectionModel();
61:         listSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
62:         siteHeaderTable.setModel(new StaticFieldTableModel(BIROFieldType.SITE_HEADER));
63:         siteHeaderTable.setDefaultRenderer(BIROField.class, new StaticFieldTableCellRenderer());
64:         siteProfileTable.setModel(new StaticFieldTableModel(BIROFieldType.SITE_PROFILE));
65:         siteProfileTable.setDefaultRenderer(BIROField.class, new StaticFieldTableCellRenderer());
66:         l =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration().getBIROFieldList();
67:         loadData();
68:
69:     }
70:
71:     /** This method is called from within the constructor to
72:      * initialize the form.
73:      * WARNING: Do NOT modify this code. The content of this method is
74:      * always regenerated by the Form Editor.
75:      */
76:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
77:     private void initComponents() {
78:
79:         siteHeaderScrollPane = new javax.swing.JScrollPane();
80:         siteHeaderTable = new javax.swing.JTable();
81:         buttonPanel = new javax.swing.JPanel();
82:         previousButton = new javax.swing.JButton();
83:         nextButton = new javax.swing.JButton();
84:         descriptionLabel = new javax.swing.JLabel();
85:         siteProfileScrollPane = new javax.swing.JScrollPane();
86:         siteProfileTable = new javax.swing.JTable();
87:         siteHeaderLabel = new javax.swing.JLabel();
88:         siteProfileLabel = new javax.swing.JLabel();
89:         jLabel1 = new javax.swing.JLabel();
```

BIROBox/src/eu/biro/birobox/panel/adaptor/DataSourceConfigurationPanel.java

```
90:         jLabel2 = new javax.swing.JLabel();
91:         jLabel3 = new javax.swing.JLabel();
92:         dataSourceNameTextField = new javax.swing.JTextField();
93:         dataSourceTypeComboBox = new javax.swing.JComboBox();
94:         jLabel4 = new javax.swing.JLabel();
95:         dataSourceIDComboBox = new javax.swing.JComboBox();
96:
97:         setBorder(javax.swing.BorderFactory.createTitledBorder("Data source configuration"));
98:
99:         siteHeaderTable.setModel(new StaticFieldTableModel(BIROFieldType.SITE_HEADER));
100:        siteHeaderTable.setColumnSelectionAllowed(true);
101:        siteHeaderTable.getTableHeader().setReorderingAllowed(false);
102:        siteHeaderScrollPane.setViewportView(siteHeaderTable);
103:
siteHeaderTable.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
104:
105:        buttonPanel.setLayout(new java.awt.GridLayout(1, 2, 10, 0));
106:
107:        previousButton.setText("Previous");
108:        previousButton.addActionListener(new java.awt.event.ActionListener() {
109:            public void actionPerformed(java.awt.event.ActionEvent evt) {
110:                previousButtonActionPerformed(evt);
111:            }
112:        });
113:        buttonPanel.add(previousButton);
114:
115:        nextButton.setText("Next");
116:        nextButton.addActionListener(new java.awt.event.ActionListener() {
117:            public void actionPerformed(java.awt.event.ActionEvent evt) {
118:                nextButtonActionPerformed(evt);
119:            }
120:        });
121:        buttonPanel.add(nextButton);
122:
123:        descriptionLabel.setText("Configure static BIRO fields");
124:
125:        siteProfileTable.setModel(new StaticFieldTableModel(BIROFieldType.SITE_PROFILE));
126:        siteProfileTable.getTableHeader().setReorderingAllowed(false);
127:        siteProfileScrollPane.setViewportView(siteProfileTable);
128:
siteProfileTable.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
129:
130:        siteHeaderLabel.setText("Site header fields");
131:
132:        siteProfileLabel.setText("Site profile fields *");
```

```
133:
134:         jLabel1.setText("Data Source Name *");
135:
136:         jLabel2.setText("Data Source ID *");
137:
138:         jLabel3.setText("Data Source Type *");
139:
140:         dataSourceTypeComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "GP", "Hospital Clinic
(Internal Medicine)", "Hospital Clinic (Diabetes)", "Regional Shared-data Register", "Regional Primary Care Project",
"Disease Management Programme", "Hospital Discharge Information", "Insurance Programme", "Retinal Screening Programme",
"Diabetes Specialist Nurse Clinic", "National Data â\200\223 Complete", "National Data â\200\223 Sample", "Regional Data
â\200\223 Sample" }));
141:
142:         jLabel4.setText("*= required fields");
143:
144:         dataSourceIDComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "1=DARTS Dataset,
Tayside, Scotland", "2=Umbria Dataset, Italy", "3=Healthgate Dataset, Austria", "4=Paulescu Datasets, Romania", "5=Norway
Diabetes Register", "6=Cyprus Diabetes Register", "7=Malta Diabetes Register" }));
145:         dataSourceIDComboBox.addActionListener(new java.awt.event.ActionListener() {
146:             public void actionPerformed(java.awt.event.ActionEvent evt) {
147:                 dataSourceIDComboBoxActionPerformed(evt);
148:             }
149:         });
150:
151:         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
152:         this.setLayout(layout);
153:         layout.setHorizontalGroup(
154:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
155:                 .addComponent(buttonPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
156:                 .addComponent(descriptionLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
157:                 .addGroup(layout.createSequentialGroup()
158:                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
159:                         .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 161, Short.MAX_VALUE)
160:                         .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
161:                         .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
162:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
163:                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
164:                         .addComponent(dataSourceTypeComboBox, 0, 319, Short.MAX_VALUE)
165:                         .addComponent(dataSourceNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 319,
Short.MAX_VALUE)
166:                         .addComponent(dataSourceIDComboBox, 0, 319, Short.MAX_VALUE)))
167:                 .addComponent(siteHeaderLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
```

```
168:         .addGroup(layout.createSequentialGroup())
169:         .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 170,
javax.swing.GroupLayout.PREFERRED_SIZE)
170:         .addContainerGap(314, Short.MAX_VALUE))
171:         .addComponent(siteHeaderScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
172:         .addComponent(siteProfileScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
173:         .addComponent(siteProfileLabel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 484, Short.MAX_VALUE)
174:     );
175:     layout.setVerticalGroup(
176:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
177:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
178:         .addComponent(descriptionLabel)
179:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
180:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
181:             .addComponent(dataSourceNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
182:             .addComponent(jLabel1))
183:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
184:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
185:             .addComponent(jLabel2)
186:             .addComponent(dataSourceIDComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
187:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
188:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
189:             .addComponent(jLabel3)
190:             .addComponent(dataSourceTypeComboBox, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE))
191:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
192:         .addComponent(siteHeaderLabel)
193:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
194:         .addComponent(siteHeaderScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 68, Short.MAX_VALUE)
195:         .addGap(18, 18, 18)
196:         .addComponent(siteProfileLabel)
197:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
198:         .addComponent(siteProfileScrollPane, javax.swing.GroupLayout.DEFAULT_SIZE, 77, Short.MAX_VALUE)
199:         .addGap(26, 26, 26)
200:         .addComponent(jLabel4)
201:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
202:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
203:     );
204:
205:     layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[] {dataSourceIDComboBox,
dataSourceNameTextField, dataSourceTypeComboBox, jLabel1, jLabel2, jLabel3});
```

```
206:
207:     }// </editor-fold>//GEN-END:initComponents
208:     private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_previousButtonActionPerformed
209:         try {
210:             rootPanel.addPanel(new ActivityTableConfigurationPanel(rootPanel));
211:             rootPanel.repaint();//GEN-LAST:event_previousButtonActionPerformed
212:         } catch (Exception ex) {
213:             Logger.getLogger(DataSourceConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
214:         }
215:     }
216:
217:     private void nextButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_nextButtonActionPerformed
218:         try {
219:             saveData();
220:             checkMapping();
221:             rootPanel.addPanel(new FieldMappingPanel(rootPanel));
222:         } catch (Exception ex) {
223:             JOptionPane.showMessageDialog(this, ex.getMessage(), "WARNING", JOptionPane.WARNING_MESSAGE);
224:             Logger.getLogger(DataSourceConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
225:         }
226:
227:
228:     }//GEN-LAST:event_nextButtonActionPerformed
229:
230:     private void dataSourceIDComboBoxActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_dataSourceIDComboBoxActionPerformed
231:         // TODO add your handling code here:
232:     }//GEN-LAST:event_dataSourceIDComboBoxActionPerformed
233:
234:     // Variables declaration - do not modify//GEN-BEGIN:variables
235:     private javax.swing.JPanel buttonPanel;
236:     private javax.swing.JComboBox dataSourceIDComboBox;
237:     private javax.swing.JTextField dataSourceNameTextField;
238:     private javax.swing.JComboBox dataSourceTypeComboBox;
239:     private javax.swing.JLabel descriptionLabel;
240:     private javax.swing.JLabel jLabel1;
241:     private javax.swing.JLabel jLabel2;
242:     private javax.swing.JLabel jLabel3;
243:     private javax.swing.JLabel jLabel4;
244:     private javax.swing.JButton nextButton;
245:     private javax.swing.JButton previousButton;
246:     private javax.swing.JLabel siteHeaderLabel;
247:     private javax.swing.JScrollPane siteHeaderScrollPane;
```

```
248: private javax.swing.JTable siteHeaderTable;
249: private javax.swing.JLabel siteProfileLabel;
250: private javax.swing.JScrollPane siteProfileScrollPane;
251: private javax.swing.JTable siteProfileTable;
252: // End of variables declaration//GEN-END:variables
253: public void loadData() {
254:     dataSourceNameTextField.setText(((String) ((StaticBIROField) l.getFieldWithName("DS_NAME")).getValue());
255:     if (((StaticBIROField) l.getFieldWithName("DS_TYPE")).getValue() == null) {
256:         dataSourceIDComboBox.setSelectedIndex(0);
257:     } else {
258:         dataSourceIDComboBox.setSelectedIndex(((Integer) ((StaticBIROField) l.getFieldWithName("DS_ID"
259:         )),getValue()) - 1);
260:     }
261:     if (((StaticBIROField) l.getFieldWithName("DS_TYPE")).getValue() == null) {
262:         dataSourceTypeComboBox.setSelectedIndex(0);
263:     } else {
264:         dataSourceTypeComboBox.setSelectedIndex(((Integer) ((StaticBIROField) l.getFieldWithName("DS_TYPE"
265:         )),getValue()) - 1);
266:     }
267: }
268: @Override
269: public void saveData() {
270:     ((StaticBIROField) l.getFieldWithName("DS_NAME")).setValue(dataSourceNameTextField.getText());
271:     ((StaticBIROField) l.getFieldWithName("DS_ID")).setValue(dataSourceIDComboBox.getSelectedIndex() + 1);
272:     ((StaticBIROField) l.getFieldWithName("DS_TYPE")).setValue(dataSourceTypeComboBox.getSelectedIndex() + 1);
273: }
274:
275: private void checkMapping() throws Exception {
276:     Iterator<BIROField> iterator = null;
277:     try {
278:         iterator = l.iterator();
279:         while (iterator.hasNext()) {
280:             BIROField b = iterator.next();
281:             if (b.getType().isSiteInfo() && b.isMandatory() && (((StaticBIROField) b).getValue().equals("") ||
282:             ((StaticBIROField) b).getValue() == null)) {
283:                 throw new Exception("Fields marked with * are required");
284:             }
285:         } catch (Exception e) {
286:             throw new Exception("Fields marked with * are required");
287:         }
288:     }
289: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      DatePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32:
33: import eu.biro.adaptor2.field.BIROField;
34: import eu.biro.adaptor2.field.DateBIROField;
35: import javax.swing.DefaultComboBoxModel;
36:
37:
38: public class DatePanel extends FieldPanel {
39:
40:     String[] datePatterns = new String[]{"dd/MM/yyyy", "dd-MM-yyyy", "yyyy-MM-dd", "MM-dd-yyyy"};
41:     private static DatePanel instance;
42:
43:     /** Creates new form DatePanel */
44:     private DatePanel() {
45:         initComponents();
```

```
46:     }
47:
48:     /** This method is called from within the constructor to
49:      * initialize the form.
50:      * WARNING: Do NOT modify this code. The content of this method is
51:      * always regenerated by the Form Editor.
52:      */
53:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
54:     private void initComponents() {
55:
56:         chooseDateLabel = new javax.swing.JLabel();
57:         dateComboBox = new javax.swing.JComboBox();
58:
59:         setLayout(new java.awt.GridLayout(2, 1));
60:
61:         chooseDateLabel.setText("Select the input date format");
62:         add(chooseDateLabel);
63:
64:         dateComboBox.setEditable(true);
65:         add(dateComboBox);
66:     } // </editor-fold>//GEN-END: initComponents
67:     @Override
68:     public void save(BIROField biroField) {
69:         DateBIROField field = (DateBIROField) biroField;
70:         field.setPattern(dateComboBox.getSelectedItem().toString());
71:     }
72:
73:     @Override
74:     public void load(BIROField biroField) {
75:         DateBIROField field = (DateBIROField) biroField;
76:         dateComboBox.setModel(new DefaultComboBoxModel(datePatterns));
77:         dateComboBox.setSelectedItem(field.getPattern());
78:     }
79:
80:     public static DatePanel getInstance() {
81:         if (instance == null) {
82:             instance = new DatePanel();
83:         }
84:         return instance;
85:     }
86:     // Variables declaration - do not modify//GEN-BEGIN:variables
87:     private javax.swing.JLabel chooseDateLabel;
88:     private javax.swing.JComboBox dateComboBox;
89:     // End of variables declaration//GEN-END:variables
90:
```

07/11/09
15:16:06

BIROBox/src/eu/biro/birobox/panel/adaptor/DatePanel.java

3

91: }

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      EnumeratedFieldMappingPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.panel.*;
33:
34: import eu.biro.adaptor2.field.BIROField;
35: import eu.biro.adaptor2.field.EnumeratedBIROField;
36: import eu.biro.adaptor2.value.ExpressionValueMapping.AllValueMapping;
37: import eu.biro.adaptor2.value.ExpressionValueMapping.AnyStringValueMapping;
38: import eu.biro.adaptor2.value.ExpressionValueMapping.NullValueMapping;
39: import eu.biro.adaptor2.value.ExpressionValueMapping.RegExpValueMapping;
40: import eu.biro.adaptor2.value.SimpleValueMapping;
41: import eu.biro.adaptor2.value.ValueMapping;
42: import java.awt.Font;
43: import java.awt.event.ActionEvent;
44: import java.awt.event.ActionListener;
45: import java.util.regex.Pattern;
```

```
46: import java.util.regex.PatternSyntaxException;
47: import javax.swing.JComboBox;
48: import javax.swing.JLabel;
49: import javax.swing.JTextField;
50: import javax.swing.SpringLayout;
51: import eu.biro.birobox.utils.SpringUtilities;
52:
53:
54: public class EnumeratedFieldMappingPanel extends FieldPanel {
55:
56:     JComboBox[] comboBoxes;
57:     String[] comboBoxesItems = new String[]{"<html>is <i>null</i></html>", "is any string", "<html>is <i>null</i>"
or any string</html>", "is regular expression", "is custom text"};
58:     JTextField[] customTextFields;
59:
60:     public EnumeratedFieldMappingPanel() {
61:         initComponents();
62:         SpringLayout layout = new SpringLayout();
63:         setLayout(layout);
64:     }
65:
66:     /** This method is called from within the constructor to
67:     * initialize the form.
68:     * WARNING: Do NOT modify this code. The content of this method is
69:     * always regenerated by the Form Editor.
70:     */
71:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
72:     private void initComponents() {
73:
74:         setMinimumSize(new java.awt.Dimension(123, 200));
75:
76:         javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
77:         this.setLayout(layout);
78:         layout.setHorizontalGroup(
79:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
80:                 .addGap(0, 123, Short.MAX_VALUE)
81:         );
82:         layout.setVerticalGroup(
83:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
84:                 .addGap(0, 200, Short.MAX_VALUE)
85:         );
86:     } // </editor-fold>//GEN-END: initComponents
87:     @Override
88:     public void save(BIROField biroField) {
89:         ValueMapping v = null;
```

```
90:         int i;
91:         EnumeratedBIROField f = (EnumeratedBIROField) biroField;
92:         ValueMapping[] valueMappingsArray = f.getValueMappings().toArray(new ValueMapping[0]);
93:
94:         for (i = 0; i < comboBoxes.length; i++) {
95:             if (comboBoxes[i].getSelectedIndex() == 0) {
96:                 v = new NullValueMapping(valueMappingsArray[i].getBIROValueName(),
valueMappingsArray[i].getDescription());
97:             } else if (comboBoxes[i].getSelectedIndex() == 1) {
98:                 v = new AnyStringValueMapping(valueMappingsArray[i].getBIROValueName(),
valueMappingsArray[i].getDescription());
99:             } else if (comboBoxes[i].getSelectedIndex() == 2) {
100:                 v = new AllValueMapping(valueMappingsArray[i].getBIROValueName(),
valueMappingsArray[i].getDescription());
101:             } else if (comboBoxes[i].getSelectedIndex() == 3) {
102:                 String pattern = customTextFields[i].getText();
103:                 try {
104:                     Pattern.compile(pattern);
105:                 } catch (PatternSyntaxException e) {
106:                     //JOptionPane.showMessageDialog(this, "This is not a valid Regular Expression", "ciao"
,JOptionPane.ERROR_MESSAGE);
107:                     pattern = "[?]";
108:                     throw e;
109:
110:                 } finally {
111:                     v = new RegExpValueMapping(valueMappingsArray[i].getBIROValueName(),
valueMappingsArray[i].getDescription(), pattern);
112:                     f.setValueMapping(v);
113:                 }
114:
115:             } else if (comboBoxes[i].getSelectedIndex() == 4) {
116:                 v = new SimpleValueMapping(valueMappingsArray[i].getBIROValueName(),
valueMappingsArray[i].getDescription(), customTextFields[i].getText());
117:
118:             }
119:             //f.removeValueMapping(valueMappingsArray[i]);
120:             f.setValueMapping(v);
121:         }
122:     }
123:
124:     @Override
125:     public void load(BIROField biroField) {
126:
127:         EnumeratedBIROField f = (EnumeratedBIROField) biroField;
128:         int i = 0;
```

```
129:         JLabel ifLabel = new JLabel();
130:
131:         comboBoxes = new JComboBox[f.getValueMappings().size()];
132:         customTextFields = new JTextField[f.getValueMappings().size()];
133:         add(new JLabel("BIROValue"));
134:         add(new JLabel(" "));
135:         add(new JLabel("Local Value"));
136:         add(new JLabel(" "));
137:         //SpringUtilities.makeCompactGrid(this, 1, 4, 0, 0, 5, 5);
138:         for (final ValueMapping v : f.getValueMappings()) {
139:             add(new JLabel(v.getDescription()));
140:             comboBoxes[i] = new JComboBox(comboBoxesItems);
141:             customTextFields[i] = new JTextField("");
142:             customTextFields[i].setEditable(false);
143:
144:             final int j = i;
145:             comboBoxes[i].addActionListener(new ActionListener() {
146:
147:                 public void actionPerformed(ActionEvent e) {
148:                     if (comboBoxes[j].getSelectedIndex() == 3) {
149:                         customTextFields[j].setEditable(true);
150:                         if (v instanceof RegExpValueMapping) {
151:                             customTextFields[j].setText(v.getDBValueName());
152:                         } else {
153:                             customTextFields[j].setText("");
154:                         }
155:                     } else if (comboBoxes[j].getSelectedIndex() == 4) {
156:                         customTextFields[j].setEditable(true);
157:                         if (v instanceof SimpleValueMapping) {
158:                             customTextFields[j].setText(v.getDBValueName());
159:                         } else {
160:                             customTextFields[j].setText("");
161:                         }
162:                     } else {
163:                         customTextFields[j].setText("");
164:                         customTextFields[j].setEditable(false);
165:                     }
166:                 }
167:             });
168:
169:
170:             if (v instanceof NullValueMapping) {
171:                 comboBoxes[i].setSelectedIndex(0);
172:                 customTextFields[i].setText("");
173:                 customTextFields[i].setEditable(false);
```

```
174:
175:     }
176:     if (v instanceof AnyStringValueMapping) {
177:         comboBoxes[i].setSelectedIndex(1);
178:         customTextFields[i].setEditable(false);
179:
180:     }
181:     if (v instanceof AllValueMapping) {
182:         comboBoxes[i].setSelectedIndex(2);
183:         customTextFields[i].setEditable(false);
184:
185:     }
186:     if (v instanceof RegExpValueMapping) {
187:         comboBoxes[i].setSelectedIndex(3);
188:         customTextFields[i].setText(v.getDBValueName());
189:         customTextFields[i].setEditable(true);
190:     }
191:     if (v instanceof SimpleValueMapping) {
192:         comboBoxes[i].setSelectedIndex(4);
193:         customTextFields[i].setText(v.getDBValueName());
194:         customTextFields[i].setEditable(true);
195:
196:     }
197:     ifLabel = new JLabel("if");
198:     add(ifLabel);
199:     ifLabel.setFont(new Font("Tahoma", Font.ITALIC, 11));
200:     add(comboBoxes[i]);
201:     add(customTextFields[i]);
202:     if ((comboBoxes[i].getSelectedIndex() == 3) | (comboBoxes[i].getSelectedIndex() == 4)) {
203:         customTextFields[i].setEditable(true);
204:     } else {
205:         customTextFields[i].setEditable(false);
206:     }
207:
208:     i++;
209: }
210: SpringUtilities.makeCompactGrid(this, (f.getValueMappings().size() + 1), 4, 0, 10, 5, 5);
211:
212: }
213: // Variables declaration - do not modify//GEN-BEGIN:variables
214: // End of variables declaration//GEN-END:variables
215: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      FieldMappingPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.birobox.panel.*;
34: import eu.biro.birobox.panel.adaptor.UnitOfMeasurementPanel;
35:
36: import eu.biro.birobox.panel.ChildrenPanel;
37: import eu.biro.birobox.panel.adaptor.DataSourceConfigurationPanel;
38: import eu.biro.birobox.panel.adaptor.DatePanel;
39: import eu.biro.birobox.panel.adaptor.EnumeratedFieldMappingPanel;
40: import eu.biro.adaptor2.BIROAdaptorConfiguration;
41: import eu.biro.adaptor2.field.BIROField;
42: import eu.biro.adaptor2.field.BIROFieldList;
43: import eu.biro.adaptor2.field.BIROFieldType;
44: import eu.biro.adaptor2.field.DateBIROField;
45: import eu.biro.adaptor2.field.EnumeratedBIROField;
```

```
46: import eu.biro.adaptor2.field.NumericBIROField;
47: import eu.biro.adaptor2.field.SimpleBIROField;
48: import java.awt.BorderLayout;
49: import java.sql.DatabaseMetaData;
50: import java.sql.ResultSet;
51: import java.sql.SQLException;
52: import java.util.Vector;
53: import java.util.logging.Level;
54: import java.util.logging.Logger;
55: import java.util.regex.PatternSyntaxException;
56: import javax.swing.JOptionPane;
57: import javax.swing.ListSelectionModel;
58: import javax.swing.event.ListSelectionEvent;
59: import javax.swing.event.ListSelectionListener;
60: import eu.biro.birobox.models.BIROFieldTableCellRenderer;
61: import eu.biro.birobox.models.BIROFieldTableModel;
62: import eu.biro.birobox.configuration.BIROBoxConfiguration;
63: import eu.biro.birobox.utils.ConnectionManager;
64: import java.util.Iterator;
65:
66:
67: public class FieldMappingPanel extends ChildrenPanel {
68:
69:     private RootPanel rootPanel;
70:     private BIROFieldList biroFieldList;
71:     private BIROField selectedValue;
72:     private FieldPanel fieldPanel;
73:     private BIROAdaptorConfiguration bIROAdaptorConfiguration;
74:     private BIROBoxConfiguration bIROBoxConfiguration;
75:     private ConnectionManager connectionManager;
76:
77:     /** Creates new form Field */
78:     public FieldMappingPanel(RootPanel rootPanel) {
79:         this.rootPanel = rootPanel;
80:         connectionManager = ConnectionManager.getManager();
81:         initComponents();
82:         ListSelectionModel listSelectionModel;
83:         listSelectionModel = fieldTable.getSelectionModel();
84:         listSelectionModel.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
85:         fieldTable.getSelectionModel().addListSelectionListener(new FieldTableListSelectionListener());
86:         fieldTable.setDefaultRenderer(BIROField.class, new BIROFieldTableCellRenderer());
87:         descriptionTextArea.setOpaque(false);
88:         bIROAdaptorConfiguration =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration();
89:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
```

```
90:     }
91:
92:     /** This method is called from within the constructor to
93:      * initialize the form.
94:      * WARNING: Do NOT modify this code. The content of this method is
95:      * always regenerated by the Form Editor.
96:      */
97:     // <editor-fold defaultstate="collapsed" desc="Generated Code">                                     );
98:     /** This method is called from within the constructor to
99:      * initialize the form.
100:      * WARNING: Do NOT modify this code. The content of this method is
101:      * always regenerated by the Form Editor.
102:      */
103:     // <editor-fold defaultstate="collapsed" desc="Generated Code"> //GEN-BEGIN: initComponents
104:     private void initComponents() {
105:
106:         jLabel1 = new javax.swing.JLabel();
107:         jScrollPane2 = new javax.swing.JScrollPane();
108:         fieldTable = new javax.swing.JTable();
109:         extractionEnableCheckBox = new javax.swing.JCheckBox();
110:         localFieldLabel = new javax.swing.JLabel();
111:         localFieldTextField = new javax.swing.JTextField();
112:         specificConfigurationPanel = new javax.swing.JPanel();
113:         jLabel2 = new javax.swing.JLabel();
114:         jPanel1 = new javax.swing.JPanel();
115:         previousButton = new javax.swing.JButton();
116:         nextButton = new javax.swing.JButton();
117:         jScrollPane1 = new javax.swing.JScrollPane();
118:         descriptionTextArea = new javax.swing.JTextArea();
119:         localFieldNameChooserButton = new javax.swing.JButton();
120:
121:         setBorder(javax.swing.BorderFactory.createTitledBorder("Fields mapping configuration"));
122:         setMaximumSize(new java.awt.Dimension(500, 400));
123:         setMinimumSize(new java.awt.Dimension(500, 400));
124:         setPreferredSize(new java.awt.Dimension(500, 400));
125:
126:         jLabel1.setText("Configure mapping between BIRO fields and local fields");
127:
128:         fieldTable.setModel(new BIROFieldTableModel());
129:         fieldTable.setColumnSelectionAllowed(true);
130:         fieldTable.getTableHeader().setReorderingAllowed(false);
131:         jScrollPane2.setViewportView(fieldTable);
132:
133:         fieldTable.getColumnModel().getSelectionModel().setSelectionMode(javax.swing.ListSelectionModel.SINGLE_SELECTION);
133:
```

BIROBox/src/eu/biro/birobox/panel/adaptor/FieldMappingPanel.java

```
134:     extractionEnableCheckBox.setText("Extract from local database");
135:     extractionEnableCheckBox.setEnabled(false);
136:     extractionEnableCheckBox.addActionListener(new java.awt.event.ActionListener() {
137:         public void actionPerformed(java.awt.event.ActionEvent evt) {
138:             extractionEnableCheckBoxActionPerformed(evt);
139:         }
140:     });
141:
142:     localFieldLabel.setText("Local field name");
143:     localFieldLabel.setEnabled(false);
144:
145:     localFieldTextField.setEnabled(false);
146:
147:     specificConfigurationPanel.setLayout(new java.awt.BorderLayout(10, 10));
148:
149:     jLabel2.setText("Description of selected BIRO field");
150:
151:     jPanel1.setLayout(new java.awt.GridLayout(1, 2, 10, 10));
152:
153:     previousButton.setText("Previous");
154:     previousButton.addActionListener(new java.awt.event.ActionListener() {
155:         public void actionPerformed(java.awt.event.ActionEvent evt) {
156:             previousButtonActionPerformed(evt);
157:         }
158:     });
159:     jPanel1.add(previousButton);
160:
161:     nextButton.setText("Next");
162:     nextButton.addActionListener(new java.awt.event.ActionListener() {
163:         public void actionPerformed(java.awt.event.ActionEvent evt) {
164:             nextButtonActionPerformed(evt);
165:         }
166:     });
167:     jPanel1.add(nextButton);
168:
169:     descriptionTextArea.setEditable(false);
170:     descriptionTextArea.setLineWrap(true);
171:     descriptionTextArea.setWrapStyleWord(true);
172:     descriptionTextArea.setBorder(null);
173:     descriptionTextArea.setOpaque(false);
174:     jScrollPane1.setViewportView(descriptionTextArea);
175:
176:     localFieldNameChooserButton.setText("...");
177:     localFieldNameChooserButton.setEnabled(false);
178:     localFieldNameChooserButton.addActionListener(new java.awt.event.ActionListener() {
```

BIROBox/src/eu/biro/birobox/panel/adaptor/FieldMappingPanel.java

```
179:         public void actionPerformed(java.awt.event.ActionEvent evt) {
180:             localFieldNameChooserButtonActionPerformed(evt);
181:         }
182:     };
183:
184:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
185:     this.setLayout(layout);
186:     layout.setHorizontalGroup(
187:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
188:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
189:             .addContainerGap()
190:             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
191:                 .addComponent(jPanel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 474, Short.MAX_VALUE)
192:                 .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 474, Short.MAX_VALUE)
193:                 .addGroup(javax.swing.GroupLayout.Alignment.LEADING, layout.createSequentialGroup()
194:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
195:                     .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 164,
javax.swing.GroupLayout.PREFERRED_SIZE)
196:                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
197:                         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
198:                             .addGap(8, 8, 8)
199:                             .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE, 302, Short.MAX_VALUE)
200:                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))
201:                         .addGroup(layout.createSequentialGroup()
202:                             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
203:                                 .addComponent(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
204:                                     .addGroup(layout.createSequentialGroup()
205:                                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
206:                                         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
207:                                             .addComponent(localFieldLabel, javax.swing.GroupLayout.PREFERRED_SIZE,
116, javax.swing.GroupLayout.PREFERRED_SIZE)
208:                                             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
209:                                                 .createSequentialGroup()
210:                                                 .addComponent(specificConfigurationPanel,
javax.swing.GroupLayout.Alignment.LEADING, javax.swing.GroupLayout.DEFAULT_SIZE, 300, Short.MAX_VALUE)
211:                                                 .addGroup(layout.createSequentialGroup()
212:                                                     .addComponent(localFieldTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 269, Short.MAX_VALUE)
213:                                                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
214:                                                     .addComponent(localFieldNameChooserButton,
```

BIROBox/src/eu/biro/birobox/panel/adaptor/FieldMappingPanel.java

```
javax.swing.GroupLayout.PREFERRED_SIZE, 25, javax.swing.GroupLayout.PREFERRED_SIZE)))
215:
.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)))
216:                                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 300,
Short.MAX_VALUE)
217:                                .addComponent(extractionEnableCheckBox, javax.swing.GroupLayout.DEFAULT_SIZE,
300, Short.MAX_VALUE))))))
218:                                .addContainerGap())
219:                                );
220:                                layout.setVerticalGroup(
221:                                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
222:                                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
223:                                .addComponent(jLabel1)
224:                                .addGap(6, 6, 6)
225:                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
226:                                .addGroup(layout.createSequentialGroup())
227:                                .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE, 20,
javax.swing.GroupLayout.PREFERRED_SIZE)
228:                                .addGap(1, 1, 1)
229:                                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 63,
javax.swing.GroupLayout.PREFERRED_SIZE)
230:                                .addGap(7, 7, 7)
231:                                .addComponent(extractionEnableCheckBox, javax.swing.GroupLayout.PREFERRED_SIZE, 18,
javax.swing.GroupLayout.PREFERRED_SIZE)
232:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
233:                                .addComponent(localFieldLabel)
234:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
235:                                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
236:                                .addComponent(localFieldTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
237:                                .addComponent(localFieldNameChooserButton))
238:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
239:                                .addComponent(specificConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 161,
Short.MAX_VALUE))
240:                                .addComponent(jScrollPane2, javax.swing.GroupLayout.DEFAULT_SIZE, 321, Short.MAX_VALUE))
241:                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
242:                                .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
243:                                );
244:                                }// </editor-fold>//GEN-END:initComponents
245:                                private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_previousButtonActionPerformed
246:                                if (selectedValue != null) {
247:                                selectedValue.setDBCColumnName(localFieldTextField.getText());
248:                                selectedValue.setRecorded(extractionEnableCheckBox.isSelected());
```

```
249:         try {
250:             if (!(selectedValue instanceof SimpleBIROField)) {
251:                 fieldPanel.save(selectedValue);
252:             }
253:         } catch (PatternSyntaxException ex) {
254:             JOptionPane.showMessageDialog(rootPanel, "This is not a valid Regular Expression", "ciao",
JOptionPane.ERROR_MESSAGE);
255:         }
256:     }
257:     try {
258:         rootPanel.addPanel(new DataSourceConfigurationPanel(rootPanel));
259:     } catch (Exception ex) {
260:         Logger.getLogger(FieldMappingPanel.class.getName()).log(Level.SEVERE, null, ex);
261:     }
262:
263: } //GEN-LAST:event_previousButtonActionPerformed
264:
265: private void nextButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_nextButtonActionPerformed
266:     if (selectedValue != null) {
267:         selectedValue.setDBCColumnName(localFieldTextField.getText());
268:         selectedValue.setRecorded(extractionEnableCheckBox.isSelected());
269:         try {
270:             if (!(selectedValue instanceof SimpleBIROField)) {
271:                 fieldPanel.save(selectedValue);
272:             }
273:         } catch (PatternSyntaxException ex) {
274:             JOptionPane.showMessageDialog(rootPanel, "This is not a valid Regular Expression", "ciao",
JOptionPane.ERROR_MESSAGE);
275:         }
276:     }
277:     try {
278:         saveData();
279:         checkMapping();
280:         rootPanel.addPanel(new SaveXMLFilePanel(rootPanel));
281:     } catch (Exception ex) {
282:         JOptionPane.showMessageDialog(rootPanel, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
283:         Logger.getLogger(FieldMappingPanel.class.getName()).log(Level.SEVERE, null, ex);
284:     }
285:
286:
287:
288: } //GEN-LAST:event_nextButtonActionPerformed
289:
290: private void extractionEnableCheckBoxActionPerformed(java.awt.event.ActionEvent evt)
```

```
{//GEN-FIRST:event_extractionEnableCheckBoxActionPerformed
291:         localFieldTextField.setEnabled(extractionEnableCheckBox.isSelected());
292:         localFieldLabel.setEnabled(extractionEnableCheckBox.isSelected());
293:         fieldPanel.enableComponents(extractionEnableCheckBox.isSelected());
294:         localFieldNameChooserButton.setEnabled(extractionEnableCheckBox.isSelected());
295:
296:     }//GEN-LAST:event_extractionEnableCheckBoxActionPerformed
297:
298:     private void localFieldNameChooserButtonActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_localFieldNameChooserButtonActionPerformed
299:         try {
300:             Object[] values = getTableFieldList().toArray();
301:             Object value = values[0];
302:             String s = (String) JOptionPane.showInputDialog(this, "Please, choose a table column", "Local field
name selection", JOptionPane.PLAIN_MESSAGE, null, values, value);
303:             if (s!=null &&!s.equals("")){
304:                 localFieldTextField.setText(s);//GEN-LAST:event_localFieldNameChooserButtonActionPerformed
305:             }
306:         } catch (Exception ex) {
307:             Logger.getLogger(FieldMappingPanel.class.getName()).log(Level.SEVERE, null, ex);
308:             JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
309:         }
310:     }
311:
312:     // Variables declaration - do not modify//GEN-BEGIN:variables
313:     private javax.swing.JTextArea descriptionTextArea;
314:     private javax.swing.JCheckBox extractionEnableCheckBox;
315:     private javax.swing.JTable fieldTable;
316:     private javax.swing.JLabel jLabel1;
317:     private javax.swing.JLabel jLabel2;
318:     private javax.swing.JPanel jPanel1;
319:     private javax.swing.JScrollPane jScrollPane1;
320:     private javax.swing.JScrollPane jScrollPane2;
321:     private javax.swing.JLabel localFieldLabel;
322:     private javax.swing.JButton localFieldNameChooserButton;
323:     private javax.swing.JTextField localFieldTextField;
324:     private javax.swing.JButton nextButton;
325:     private javax.swing.JButton previousButton;
326:     private javax.swing.JPanel specificConfigurationPanel;
327:     // End of variables declaration//GEN-END:variables
328:
329:
330: private class FieldTableListSelectionListener implements ListSelectionListener {
331:
332:     public FieldTableListSelectionListener() {
```

```
333:
334:     }
335:
336:     public void valueChanged(ListSelectionEvent e) {
337:
338:         if (selectedValue != null) {
339:             selectedValue.setDBCColumnName(localFieldTextField.getText());
340:             selectedValue.setRecorded(extractionEnableCheckBox.isSelected());
341:
342:             try {
343:                 if (!(selectedValue instanceof SimpleBIROField)) {
344:                     fieldPanel.save(selectedValue);
345:                 }
346:             } catch (PatternSyntaxException ex) {
347:                 JOptionPane.showMessageDialog(rootPanel, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
348:             }
349:         }
350:     }
351:
352:     if (!e.getValueIsAdjusting()) {
353:         selectedValue = (BIROField) fieldTable.getValueAt(fieldTable.getSelectedRow(), 0);
354:         localFieldTextField.setText(selectedValue.getDBCColumnName());
355:
356:         extractionEnableCheckBox.setSelected(selectedValue.isRecorded() || selectedValue.isMandatory());
357:         extractionEnableCheckBox.setEnabled(!selectedValue.isMandatory());
358:
359:         if (selectedValue.getType().name().equalsIgnoreCase("ACTIVITY_DATA") &&
!bIROAdaptorConfiguration.isActivityTableEnabled()) {
360:             extractionEnableCheckBox.setSelected(false);
361:             extractionEnableCheckBox.setEnabled(false);
362:         }
363:
364:         localFieldLabel.setEnabled(extractionEnableCheckBox.isSelected());
365:         localFieldTextField.setEnabled(extractionEnableCheckBox.isSelected());
366:         localFieldNameChooserButton.setEnabled(extractionEnableCheckBox.isSelected());
367:
368:         specificConfigurationPanel.invalidate();
369:         specificConfigurationPanel.removeAll();
370:         if (!(selectedValue instanceof SimpleBIROField)) {
371:             if (selectedValue instanceof NumericBIROField) {
372:                 fieldPanel = UnitOfMeasurementPanel.getInstance();
373:             } else if (selectedValue instanceof DateBIROField) {
374:                 fieldPanel = DatePanel.getInstance();
375:             } else if (selectedValue instanceof EnumeratedBIROField) {
376:                 fieldPanel = new EnumeratedFieldMappingPanel();
```

```
377:         }
378:         fieldPanel.load(selectedValue);
379:         specificConfigurationPanel.add(fieldPanel, BorderLayout.NORTH);
380:         fieldPanel.enableComponents(extractionEnableCheckBox.isSelected());
381:     }
382:     descriptionTextArea.setText(selectedValue.getDescription());
383:     specificConfigurationPanel.revalidate();
384:     specificConfigurationPanel.repaint();
385: }
386:
387: }
388:
389: }
390:
391: @Override
392: public void saveData() {
393:     if (selectedValue != null) {
394:         selectedValue.setDBCColumnName(localFieldTextField.getText());
395:         selectedValue.setRecorded(extractionEnableCheckBox.isSelected());
396:         try {
397:             if (!(selectedValue instanceof SimpleBIROField)) {
398:                 fieldPanel.save(selectedValue);
399:             }
400:         } catch (PatternSyntaxException ex) {
401:             JOptionPane.showMessageDialog(rootPanel, "This is not a valid Regular Expression", "Warning",
JOptionPane.ERROR_MESSAGE);
402:         }
403:     }
404:
405:
406: }
407:
408: private Vector<String> getTableFieldList() throws SQLException, ClassNotFoundException, Exception {
409:
410:     Vector<String> fields = new Vector<String>();
411:     fields = new Vector<String>();
412:     String tableName;
413:     connectionManager.connect();
414:     DatabaseMetaData md = connectionManager.getMetaData();
415:
416:     if (selectedValue.getType().name().equalsIgnoreCase("ACTIVITY_DATA")) {
417:         tableName = bIROAdaptorConfiguration.getActivityTableName();
418:     } else {
419:         tableName = bIROAdaptorConfiguration.getMergeTableName();
420:     }
```

```
421:         ResultSet rs = md.getColumns(null, null, tableName, null);
422:
423:         while (rs.next()) {
424:             fields.add(rs.getString("COLUMN_NAME"));
425:         }
426:         return fields;
427:
428:     }
429: }
430: private void checkMapping() throws Exception{
431:     Iterator<BIROField> iterator = bIROAdaptorConfiguration.getBIROFieldList().iterator();
432:     while(iterator.hasNext())
433:     {
434:         BIROField b = iterator.next();
435:         if (b.isRecorded() && !b.getType().isSiteInfo() && (b.getDBColumnName().equals("")
)| b.getDBColumnName() == null))
436:         {
437:             throw new Exception("Some BIRO fields have an invalid mapping.\nPlease check local fields' names");
438:         }
439:     }
440: }
441: }
442:
443:
444:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      FieldPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.adaptor2.field.BIROField;
33: import java.awt.Component;
34: import javax.swing.JPanel;
35:
36:
37: public abstract class FieldPanel extends JPanel {
38:
39:     public abstract void save(BIROField biroField);
40:
41:     public abstract void load(BIROField biroField);
42:
43:     public void enableComponents(boolean enabled) {
44:         Component[] components = this.getComponents();
45:         for (int i = 0; i < components.length; i++) {
```

```
46:         components[i].setEnabled(enabled);
47:     }
48: }
49: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      MergeTableConfigurationPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.birobox.utils.ConnectionManager;
34: import eu.biro.birobox.configuration.BIROBoxConfiguration;
35: import eu.biro.birobox.panel.*;
36: import eu.biro.birobox.main.*;
37: import eu.biro.birobox.panel.ChildrenPanel;
38: import eu.biro.birobox.panel.adaptor.DataSourceConfigurationPanel;
39: import eu.biro.adaptor2.BIROAdaptorConfiguration;
40: import eu.biro.adaptor2.Drivers.CSVFILEDriver;
41: import java.io.IOException;
42: import java.sql.DatabaseMetaData;
43: import java.sql.ResultSet;
44: import java.sql.SQLException;
45: import java.text.SimpleDateFormat;
```

```
46: import java.util.Vector;
47: import java.util.logging.Level;
48: import java.util.logging.Logger;
49: import javax.swing.JOptionPane;
50:
51: public class MergeTableConfigurationPanel extends ChildrenPanel {
52:
53:     private RootPanel rootPanel;
54:     BIROAdaptorConfiguration bIROAdaptorConfiguration;
55:     BIROBoxConfiguration bIROBoxConfiguration;
56:     private ConnectionManager connectionManager;
57:     private SimpleDateFormat formatter;
58:
59:     /** Creates new form NewJPanel */
60:     public MergeTableConfigurationPanel(RootPanel rootPanel) throws IOException {
61:         this.rootPanel = rootPanel;
62:         connectionManager = ConnectionManager.getManager();
63:         initComponents();
64:         formatter = new SimpleDateFormat("dd/MM/yyyy");
65:         startingDatePicker.setFormats(formatter);
66:         endingDatePicker.setFormats(formatter);
67:         loadData();
68:
69:         episodeDateColumnNameTextField.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
70:         episodeDateColumnNameLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
71:         episodeDateColumnNameButton.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
72:         startingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
73:         startingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
74:         endingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
75:         endingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
76:     }
77:
78:     /** This method is called from within the constructor to
79:      * initialize the form.
80:      * WARNING: Do NOT modify this code. The content of this method is
81:      * always regenerated by the Form Editor.
82:      */
83:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
```

```
84: private void initComponents() {
85:
86:     buttonGroup1 = new javax.swing.ButtonGroup();
87:     MergeTableConfigurationPanel = new javax.swing.JPanel();
88:     descriptionLabel = new javax.swing.JLabel();
89:     buttonPanel = new javax.swing.JPanel();
90:     previousButton = new javax.swing.JButton();
91:     nextButton = new javax.swing.JButton();
92:     mergeTableCreationRadioButton = new javax.swing.JRadioButton();
93:     mergeTableSelectionRadioButton = new javax.swing.JRadioButton();
94:     jScrollPane1 = new javax.swing.JScrollPane();
95:     mergeTableCreationTextArea = new javax.swing.JTextArea();
96:     mergeTableSelectionButton = new javax.swing.JButton();
97:     mergeTableSelectionTextField = new javax.swing.JTextField();
98:     startingDateLabel = new javax.swing.JLabel();
99:     endingDateLabel = new javax.swing.JLabel();
100:    endingDatePicker = new org.jdesktop.swing.JXDatePicker();
101:    startingDatePicker = new org.jdesktop.swing.JXDatePicker();
102:    queryConditionCheckBox = new javax.swing.JCheckBox();
103:    episodeDateColumnNameLabel = new javax.swing.JLabel();
104:    episodeDateColumnNameTextField = new javax.swing.JTextField();
105:    episodeDateColumnNameButton = new javax.swing.JButton();
106:
107:    MergeTableConfigurationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Mergetable
configuration"));
108:
109:    descriptionLabel.setText("Configure the mergetable ");
110:
111:    buttonPanel.setMinimumSize(new java.awt.Dimension(0, 0));
112:    buttonPanel.setLayout(new java.awt.GridLayout(1, 2, 5, 0));
113:
114:    previousButton.setText("Previous");
115:    previousButton.addActionListener(new java.awt.event.ActionListener() {
116:        public void actionPerformed(java.awt.event.ActionEvent evt) {
117:            previousButtonActionPerformed(evt);
118:        }
119:    });
120:    buttonPanel.add(previousButton);
121:
122:    nextButton.setText("Next");
123:    nextButton.addActionListener(new java.awt.event.ActionListener() {
124:        public void actionPerformed(java.awt.event.ActionEvent evt) {
125:            nextButtonActionPerformed(evt);
126:        }
127:    });
```

BIROBox/src/eu/biro/birobox/panel/adaptor/MergeTableConfigurationPanel.java

```
128:         buttonPanel.add(nextButton);
129:
130:         buttonGroup1.add(mergeTableCreationRadioButton);
131:         mergeTableCreationRadioButton.setText("select the mergetable from a custom query");
132:         mergeTableCreationRadioButton.addChangeListener(new javax.swing.event.ChangeListener() {
133:             public void stateChanged(javax.swing.event.ChangeEvent evt) {
134:                 mergeTableCreationRadioButtonStateChanged(evt);
135:             }
136:         });
137:
138:         buttonGroup1.add(mergeTableSelectionRadioButton);
139:         mergeTableSelectionRadioButton.setSelected(true);
140:         mergeTableSelectionRadioButton.setText("select the mergetable from already present table");
141:         mergeTableSelectionRadioButton.addChangeListener(new javax.swing.event.ChangeListener() {
142:             public void stateChanged(javax.swing.event.ChangeEvent evt) {
143:                 mergeTableSelectionRadioButtonStateChanged(evt);
144:             }
145:         });
146:
147:         mergeTableCreationTextArea.setColumns(20);
148:         mergeTableCreationTextArea.setLineWrap(true);
149:         mergeTableCreationTextArea.setRows(5);
150:         mergeTableCreationTextArea.setEnabled(false);
151:         mergeTableCreationTextArea.setOpaque(false);
152:         jScrollPane1.setViewportViewView(mergeTableCreationTextArea);
153:
154:         mergeTableSelectionButton.setText("...");
155:         mergeTableSelectionButton.addActionListener(new java.awt.event.ActionListener() {
156:             public void actionPerformed(java.awt.event.ActionEvent evt) {
157:                 mergeTableSelectionButtonActionPerformed(evt);
158:             }
159:         });
160:
161:         startingDateLabel.setText("starting date for episodes");
162:
163:         endingDateLabel.setText("ending date for episodes");
164:
165:         queryConditionCheckBox.setText("select a subset of episode dates (only for database)");
166:         queryConditionCheckBox.addItemListener(new java.awt.event.ItemListener() {
167:             public void itemStateChanged(java.awt.event.ItemEvent evt) {
168:                 queryConditionCheckBoxItemStateChanged(evt);
169:             }
170:         });
171:
172:         episodeDateColumnNameLabel.setText("episode date column name");
```

```

173:         episodeDateColumnNameButton.setText("...");
174:         episodeDateColumnNameButton.addActionListener(new java.awt.event.ActionListener() {
175:             public void actionPerformed(java.awt.event.ActionEvent evt) {
176:                 episodeDateColumnNameButtonActionPerformed(evt);
177:             }
178:         });
179:     });
180:
181:     javax.swing.GroupLayout MergeTableConfigurationPanelLayout = new
javax.swing.GroupLayout(MergeTableConfigurationPanel);
182:     MergeTableConfigurationPanel.setLayout(MergeTableConfigurationPanelLayout);
183:     MergeTableConfigurationPanelLayout.setHorizontalGroup(
184:         MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
185:         .addGroup(MergeTableConfigurationPanelLayout.createSequentialGroup())
186:         .addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
187:             .addComponent(descriptionLabel)
188:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
MergeTableConfigurationPanelLayout.createSequentialGroup())
189:             .addComponent(mergeTableSelectionTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 464,
Short.MAX_VALUE)
190:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
191:             .addComponent(mergeTableSelectionButton, javax.swing.GroupLayout.PREFERRED_SIZE, 55,
javax.swing.GroupLayout.PREFERRED_SIZE)
192:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED))
193:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
MergeTableConfigurationPanelLayout.createSequentialGroup())
194:             .addGap(45, 45, 45)
195:             .addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
196:                 .addComponent(queryConditionCheckBox)
197:                 .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
MergeTableConfigurationPanelLayout.createSequentialGroup())
198:                 .addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
199:                     .addComponent(endingDateLabel)
200:                     .addComponent(episodeDateColumnNameLabel)
201:                     .addComponent(startingDateLabel))
202:                 .addGap(54, 54, 54)
203:                 .addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
204:                     .addGroup(MergeTableConfigurationPanelLayout.createSequentialGroup())
205:                     .addComponent(episodeDateColumnNameTextField,
javax.swing.GroupLayout.DEFAULT_SIZE, 238, Short.MAX_VALUE)
206:                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```
207:                .addComponent(episodeDateColumnNameButton,
javax.swing.GroupLayout.PREFERRED_SIZE, 55, javax.swing.GroupLayout.PREFERRED_SIZE))
208:                .addComponent(startingDatePicker, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 299, Short.MAX_VALUE)
209:                .addComponent(endingDatePicker, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 299, Short.MAX_VALUE))))))
210:                .addContainerGap()
211:                .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 525, Short.MAX_VALUE)
212:                .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 525, Short.MAX_VALUE)
213:                .addGroup(MergeTableConfigurationPanelLayout.createSequentialGroup()
214:                .addComponent(mergeTableCreationRadioButton, javax.swing.GroupLayout.PREFERRED_SIZE, 271,
javax.swing.GroupLayout.PREFERRED_SIZE)
215:                .addContainerGap()
216:                .addGroup(MergeTableConfigurationPanelLayout.createSequentialGroup()
217:                .addComponent(mergeTableSelectionRadioButton)
218:                .addContainerGap()
219:                );
220:                MergeTableConfigurationPanelLayout.setVerticalGroup(
221:                MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
222:                .addGroup(MergeTableConfigurationPanelLayout.createSequentialGroup()
223:                .addComponent(descriptionLabel)
224:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
225:                .addComponent(mergeTableSelectionRadioButton)
226:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
227:
.addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
228:                .addComponent(mergeTableSelectionButton)
229:                .addComponent(mergeTableSelectionTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
230:                .addGap(16, 16, 16)
231:                .addComponent(queryConditionCheckBox)
232:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
233:
.addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
234:                .addComponent(episodeDateColumnNameLabel)
235:                .addComponent(episodeDateColumnNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
236:                .addComponent(episodeDateColumnNameButton))
237:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
238:
.addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
239:                .addComponent(startingDateLabel)
240:                .addComponent(startingDatePicker, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
241:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
242:
.addGroup(MergeTableConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
243:         .addComponent(endingDateLabel)
244:         .addComponent(endingDatePicker, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
245:         .addGap(18, 18, 18)
246:         .addComponent(mergeTableCreationRadioButton)
247:         .addGap(7, 7, 7)
248:         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
249:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 179, Short.MAX_VALUE)
250:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
251:    );
252:
253:    MergeTableConfigurationPanelLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{endingDateLabel, endingDatePicker, episodeDateColumnNameButton, episodeDateColumnNameLabel,
episodeDateColumnNameTextField, queryConditionCheckBox, startingDateLabel, startingDatePicker});
254:
255:    MergeTableConfigurationPanelLayout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{mergeTableSelectionButton, mergeTableSelectionTextField});
256:
257:    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
258:    this.setLayout(layout);
259:    layout.setHorizontalGroup(
260:        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
261:        .addComponent(MergeTableConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
262:    );
263:    layout.setVerticalGroup(
264:        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
265:        .addComponent(MergeTableConfigurationPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
266:    );
267:    } // </editor-fold> // GEN-END: initComponents
268:    private void nextButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_nextButtonActionPerformed
269:        try {
270:            saveData();
271:            checkConnection();
272:            rootPanel.addPanel(new ActivityTableConfigurationPanel(rootPanel));
273:        } catch (Exception ex) {
274:            JOptionPane.showMessageDialog(this, ex.getMessage(), "WARNING", JOptionPane.WARNING_MESSAGE);
275:            Logger.getLogger(MergeTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
276:        }
```

BIROBox/src/eu/biro/birobox/panel/adaptor/MergeTableConfigurationPanel.java

```
277:         } //GEN-LAST:event_nextButtonActionPerformed
278:
279:         private void mergeTableCreationRadioButtonStateChanged( javax.swing.event.ChangeEvent evt )
{ //GEN-FIRST:event_mergeTableCreationRadioButtonStateChanged
280:             mergeTableCreationTextArea.setEnabled(mergeTableCreationRadioButton.isSelected());
281:             mergeTableCreationTextArea.setOpaque(mergeTableCreationRadioButton.isSelected());
282:             /*
283:             episodeDateColumnNameTextField.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
284:             episodeDateColumnNameLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
285:             episodeDateColumnNameButton.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
286:             startingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
287:             startingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
288:             endingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
289:             endingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
290:             * */
291:
292:         } //GEN-LAST:event_mergeTableCreationRadioButtonStateChanged
293:
294:         private void mergeTableSelectionRadioButtonStateChanged( javax.swing.event.ChangeEvent evt )
{ //GEN-FIRST:event_mergeTableSelectionRadioButtonStateChanged
295:             mergeTableSelectionTextField.setEnabled(mergeTableSelectionRadioButton.isSelected());
296:             mergeTableSelectionButton.setEnabled(mergeTableSelectionRadioButton.isSelected());
297:             queryConditionCheckBox.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
!(bIROAdaptorConfiguration.getDBMSDriver() instanceof CSVFILEDriver));
298:
299:             episodeDateColumnNameTextField.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
300:             episodeDateColumnNameLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
301:             episodeDateColumnNameButton.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
302:             startingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
303:             startingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
304:             endingDatePicker.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
305:             endingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
```

```
queryConditionCheckBox.isSelected());
306:     } //GEN-LAST:event_mergeTableSelectionRadioButtonStateChanged
307:
308:     private void mergeTableSelectionButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_mergeTableSelectionButtonActionPerformed
309:         try {
310:             Object[] values = getTablesList().toArray();
311:             Object value = values[0];
312:             String s = (String) JOptionPane.showInputDialog(this, "Please, select a table", "Merge Table name
selection", JOptionPane.PLAIN_MESSAGE, null, values, value);
313:             if (s != null) {
314:                 mergeTableSelectionTextField.setText(s);
315:             }
316:         } catch (Exception ex) {
317:             JOptionPane.showMessageDialog(this, ex.getMessage(), "Warning", JOptionPane.WARNING_MESSAGE);
318:             Logger.getLogger(MergeTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
319:         }
320:
321:     } //GEN-LAST:event_mergeTableSelectionButtonActionPerformed
322:
323:     private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_previousButtonActionPerformed
324:         try {
325:             rootPanel.addPanel(new ConnectionConfigurationPanel(rootPanel));
326:         } catch (Exception ex) {
327:             Logger.getLogger(MergeTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
328:         }
329:     } //GEN-LAST:event_previousButtonActionPerformed
330:
331:     private void episodeDateColumnNameButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_episodeDateColumnNameButtonActionPerformed
332:         try {
333:             Object[] values = getMergeTableFieldList().toArray();
334:             Object value = values[0];
335:             String s = (String) JOptionPane.showInputDialog(this, "Please, choose the episode date column",
"Episode date column selection", JOptionPane.PLAIN_MESSAGE, null, values, value);
336:             episodeDateColumnNameTextField.setText(s);
337:         } catch (Exception ex) {
338:             Logger.getLogger(FieldMappingPanel.class.getName()).log(Level.SEVERE, null, ex);
339:             JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
340:         }
341:     } //GEN-LAST:event_episodeDateColumnNameButtonActionPerformed
342:
343:     private void queryConditionCheckBoxItemStateChanged(java.awt.event.ItemEvent evt)
{ //GEN-FIRST:event_queryConditionCheckBoxItemStateChanged
```

```
344:
345:         episodeDateColumnNameTextField.setEnabled(queryConditionCheckBox.isSelected());
346:         episodeDateColumnNameButton.setEnabled(queryConditionCheckBox.isSelected());
347:         startingDatePicker.setEnabled(queryConditionCheckBox.isSelected());
348:         endingDatePicker.setEnabled(queryConditionCheckBox.isSelected());
349:         endingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
350:         startingDateLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
351:         episodeDateColumnNameLabel.setEnabled(mergeTableSelectionRadioButton.isSelected() &&
queryConditionCheckBox.isSelected());
352:
353:     } //GEN-LAST:event_queryConditionCheckBoxItemStateChanged
354:
355:     private void loadData() throws IOException {
356:         bIROAdaptorConfiguration =
(BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList()).getCurrentConfiguration();
357:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
358:         mergeTableSelectionRadioButton.setSelected(bIROAdaptorConfiguration.isMergeTableSelected());
359:         mergeTableCreationRadioButton.setSelected(!bIROAdaptorConfiguration.isMergeTableSelected());
360:         mergeTableSelectionTextField.setText(bIROAdaptorConfiguration.getMergeTableName());
361:         mergeTableCreationTextArea.setText(bIROAdaptorConfiguration.getMergeTableCreationQuery());
362:         if (bIROAdaptorConfiguration.getDBMSDriver() instanceof CSVFILEDriver) {
363:             queryConditionCheckBox.setEnabled(false);
364:             queryConditionCheckBox.setSelected(false);
365:         } else {
366:             queryConditionCheckBox.setEnabled(true);
367:             queryConditionCheckBox.setSelected(bIROAdaptorConfiguration.isQueryConditionSelected());
368:             episodeDateColumnNameTextField.setText(bIROAdaptorConfiguration.getEpisodeDateColumnName());
369:             startingDatePicker.setDate(bIROAdaptorConfiguration.getEpisodeStartingDate());
370:             endingDatePicker.setDate(bIROAdaptorConfiguration.getEpisodeEndingDate());
371:         }
372:
373:     }
374:
375:     public void saveData() throws IOException {
376:         bIROAdaptorConfiguration.setMergeTableSelected(mergeTableSelectionRadioButton.isSelected());
377:         bIROAdaptorConfiguration.setMergeTableName(mergeTableSelectionTextField.getText());
378:         bIROAdaptorConfiguration.setMergeTableCreationQuery(mergeTableCreationTextArea.getText());
379:         bIROAdaptorConfiguration.setQueryConditionSelected(queryConditionCheckBox.isSelected());
380:         bIROAdaptorConfiguration.setEpisodeDateColumnName(episodeDateColumnNameTextField.getText());
381:         bIROAdaptorConfiguration.setEpisodeStartingDate(startingDatePicker.getDate());
382:         bIROAdaptorConfiguration.setEpisodeEndingDate(endingDatePicker.getDate());
383:
384:         if (mergeTableSelectionRadioButton.isSelected()) {
```

BIROBox/src/eu/biro/birobox/panel/adaptor/MergeTableConfigurationPanel.java

```
385:         String mergeTableQuery = "SELECT * FROM \"" + mergeTableSelectionTextField.getText()+"\"";
386:         if (queryConditionCheckBox.isSelected() && !bIROAdaptorConfiguration.getEpisodeDateColumnName().equals(
"")) {
387:             mergeTableQuery = mergeTableQuery + " WHERE \"" +
bIROAdaptorConfiguration.getEpisodeDateColumnName() + "\" > '" +
formatter.format(bIROAdaptorConfiguration.getEpisodeStartingDate()) + "' AND \"" +
bIROAdaptorConfiguration.getEpisodeDateColumnName() + "\" < '" +
formatter.format(bIROAdaptorConfiguration.getEpisodeEndingDate()) + "'";
388:         }
389:
390:         bIROAdaptorConfiguration.setMergeTableQuery(mergeTableQuery);
391:         //System.out.println(bIROAdaptorConfiguration.getMergeTableQuery());
392:
393:     } else {
394:         bIROAdaptorConfiguration.setMergeTableQuery(mergeTableCreationTextArea.getText());
395:     }
396:
397: }
398:
399: private Vector<String> getTablesList() throws SQLException, ClassNotFoundException, Exception {
400:     Vector<String> tables = new Vector<String>();
401:     connectionManager.connect();
402:     DatabaseMetaData md = connectionManager.getMetaData();
403:
404:     ResultSet rs = md.getTables(null, null, "%", new String[]{"TABLE"});
405:     while (rs.next()) {
406:         //System.out.println(rs.getString(3));
407:         tables.add(rs.getString(3));
408:     }
409:     System.out.println(md.getDatabaseProductName());
410:     System.out.println(md.getDriverName());
411:
412:     connectionManager.close();
413:     return tables;
414:
415: }
416:
417: private Vector<String> getMergeTableFieldList() throws SQLException, ClassNotFoundException, Exception {
418:     Vector<String> fields = new Vector<String>();
419:     fields = new Vector<String>();
420:     connectionManager.connect();
421:     DatabaseMetaData md = connectionManager.getMetaData();
422:     /*ResultSet rs = md.getTables(null, null, "%", new String[]{"TABLE"});*/
423:     ResultSet rs = md.getColumns(null, null, mergeTableSelectionTextField.getText(), null);
424:
```

```
425:         while (rs.next()) {
426:             //System.out.println(rs.getString("COLUMN_NAME"));
427:             fields.add(rs.getString("COLUMN_NAME"));
428:         }
429:         return fields;
430:     }
431:
432:     private void checkConnection() throws Exception {
433:         try {
434:             ConnectionManager manager =
ConnectionManager.initializeManager(bIROAdaptorConfiguration.getDBMSDriver());
435:             manager.connect();
436:             ResultSet r = manager.executeQuery(bIROAdaptorConfiguration.getMergeTableQuery());
437:             if (!r.next()) {
438:                 throw new Exception("The query returned an empty result set");
439:             }
440:         } catch (Exception ex) {
441:             Logger.getLogger(MergeTableConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
442:             throw new Exception("An error occurred when attempting to establish a connection to the merge table.\n"
+ ex.getMessage());
443:         }
444:     }
445:     // Variables declaration - do not modify//GEN-BEGIN:variables
446:     private javax.swing.JPanel MergeTableConfigurationPanel;
447:     private javax.swing.ButtonGroup buttonGroup1;
448:     private javax.swing.JPanel buttonPanel;
449:     private javax.swing.JLabel descriptionLabel;
450:     private javax.swing.JLabel endingDateLabel;
451:     private org.jdesktop.swing.JXDatePicker endingDatePicker;
452:     private javax.swing.JButton episodeDateColumnNameButton;
453:     private javax.swing.JLabel episodeDateColumnNameLabel;
454:     private javax.swing.JTextField episodeDateColumnNameTextField;
455:     private javax.swing.JScrollPane jScrollPane1;
456:     private javax.swing.JRadioButton mergeTableCreationRadioButton;
457:     private javax.swing.JTextArea mergeTableCreationTextArea;
458:     private javax.swing.JButton mergeTableSelectionButton;
459:     private javax.swing.JRadioButton mergeTableSelectionRadioButton;
460:     private javax.swing.JTextField mergeTableSelectionTextField;
461:     private javax.swing.JButton nextButton;
462:     private javax.swing.JButton previousButton;
463:     private javax.swing.JCheckBox queryConditionCheckBox;
464:     private javax.swing.JLabel startingDateLabel;
465:     private org.jdesktop.swing.JXDatePicker startingDatePicker;
466:     // End of variables declaration//GEN-END:variables
467: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      SaveXMLFilePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
33: import eu.biro.birobox.configuration.BIROBoxConfiguration;
34: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
35: import eu.biro.birobox.panel.ChildrenPanel;
36: import eu.biro.adaptor2.BIROAdaptorConfiguration;
37: import eu.biro.birobox.configuration.Configuration;
38: import java.awt.Cursor;
39: import java.io.File;
40: import java.util.logging.Level;
41: import java.util.logging.Logger;
42: import javax.swing.JFileChooser;
43: import javax.swing.JOptionPane;
44: import javax.swing.filechooser.FileFilter;
45: import javax.swing.filechooser.FileNameExtensionFilter;
```

```
46: import eu.biro.birobox.panel.RootPanel;
47: import java.text.DateFormat;
48: import java.text.SimpleDateFormat;
49: import java.util.Calendar;
50: import java.util.Date;
51: import javax.swing.SwingUtilities;
52:
53: public class SaveXMLFilePanel extends ChildrenPanel {
54:
55:     private RootPanel rootPanel;
56:     private BIROAdaptorConfiguration bIROAdaptorConfiguration;
57:     private BIROAdaptorConfigurationList bIROAdaptorConfigurationList;
58:     private BIROBoxConfiguration bIROBoxConfiguration;
59:     private BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
60:
61:     /** Creates new form SaveXMLFilePanel */
62:     public SaveXMLFilePanel(RootPanel rootPanel) {
63:         this.rootPanel = rootPanel;
64:         initComponents();
65:         bIROAdaptorConfigurationList = BIROAdaptorConfigurationList.getBIROAdaptorConfigurationList();
66:         bIROAdaptorConfiguration = bIROAdaptorConfigurationList.getCurrentConfiguration();
67:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
68:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
69:         fileNameTextField.setText(bIROAdaptorConfiguration.getExportFilePath());
70:
71:         if (bIROAdaptorConfiguration.getExportFilePath().equals("")) {
72:             DateFormat dateFormat = new SimpleDateFormat("ddMMyyHHmmss");
73:             Date date = new Date();
74:             fileNameTextField.setText(Configuration.root + Configuration.bIROAdaptorDefaultOutputFolder + ".xml" +
dateFormat.format(date) + ".zip");
75:         }
76:
77:     }
78:
79:     /** This method is called from within the constructor to
80:     * initialize the form.
81:     * WARNING: Do NOT modify this code. The content of this method is
82:     * always regenerated by the Form Editor.
83:     */
84:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
85:     private void initComponents() {
86:
87:         jLabel1 = new javax.swing.JLabel();
88:         fileNameTextField = new javax.swing.JTextField();
89:         browseButton = new javax.swing.JButton();
```

```
90:         buttonPanel = new javax.swing.JPanel();
91:         previousButton = new javax.swing.JButton();
92:         closeButton = new javax.swing.JButton();
93:
94:         setBorder(javax.swing.BorderFactory.createTitledBorder("Output file configuration"));
95:
96:         jLabel1.setText("Configure the output archive file name");
97:
98:         fileNameTextField.setEditable(false);
99:
100:        browseButton.setText("Browse");
101:        browseButton.addActionListener(new java.awt.event.ActionListener() {
102:            public void actionPerformed(java.awt.event.ActionEvent evt) {
103:                browseButtonActionPerformed(evt);
104:            }
105:        });
106:
107:        buttonPanel.setLayout(new java.awt.GridLayout(1, 0));
108:
109:        previousButton.setText("Previous");
110:        previousButton.addActionListener(new java.awt.event.ActionListener() {
111:            public void actionPerformed(java.awt.event.ActionEvent evt) {
112:                previousButtonActionPerformed(evt);
113:            }
114:        });
115:        buttonPanel.add(previousButton);
116:
117:        closeButton.setText("Close");
118:        closeButton.addActionListener(new java.awt.event.ActionListener() {
119:            public void actionPerformed(java.awt.event.ActionEvent evt) {
120:                closeButtonActionPerformed(evt);
121:            }
122:        });
123:        buttonPanel.add(closeButton);
124:
125:        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
126:        this.setLayout(layout);
127:        layout.setHorizontalGroup(
128:            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
129:                .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 384, Short.MAX_VALUE)
130:                .addGroup(layout.createSequentialGroup()
131:                    .addComponent(fileNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 311, Short.MAX_VALUE)
132:                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
133:                    .addComponent(browseButton))
134:                .addComponent(buttonPanel, javax.swing.GroupLayout.Alignment.TRAILING,
```

BIROBox/src/eu/biro/birobox/panel/adaptor/SaveXMLFilePanel.java

```
javax.swing.GroupLayout.DEFAULT_SIZE, 384, Short.MAX_VALUE)
135:         );
136:         layout.setVerticalGroup(
137:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
138:             .addGroup(layout.createSequentialGroup()
139:                 .addComponent(jLabel1)
140:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
141:                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
142:                     .addComponent(fileNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
143:                     .addComponent(browseButton))
144:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 351, Short.MAX_VALUE)
145:                 .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
146:         );
147:     } // </editor-fold> // GEN-END: initComponents
148:     private void previousButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_previousButtonActionPerformed
149:         try {
150:             bIROAdaptorConfiguration.setExportFilePath(fileNameTextField.getText());
151:             rootPanel.addPanel(new FieldMappingPanel(rootPanel));
152:         } catch (Exception ex) {
153:             Logger.getLogger(SaveXMLFilePanel.class.getName()).log(Level.SEVERE, null, ex);
154:         }
155:     } // GEN-LAST: event_previousButtonActionPerformed
156:
157:     private void browseButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_browseButtonActionPerformed
158:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
159:
160:         SwingUtilities.invokeLater(new Runnable() {
161:
162:             public void run() {
163:                 boolean selectionFinished = false;
164:                 JFileChooser fc = new JFileChooser(new File(Configuration.root));
165:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("File ZIP", "zip"));
166:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
167:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
168:                 while (!selectionFinished) {
169:                     int returnVal = fc.showSaveDialog(SaveXMLFilePanel.this);
170:                     if (returnVal == JFileChooser.APPROVE_OPTION) {
171:                         File file = fc.getSelectedFile();
172:                         if (file.exists()) {
173:                             int n = JOptionPane.showConfirmDialog(SaveXMLFilePanel.this, "Do you want to overwrite
the selected file?", "Warning", JOptionPane.YES_NO_OPTION);
```

```
174:             if (n == 0) {
175:                 fileNameTextField.setText(file.getAbsolutePath());
176:                 selectionFinished = true;
177:             }
178:         } else {
179:             fileNameTextField.setText(file.getAbsolutePath() + ".zip");
180:             selectionFinished = true;
181:         }
182:     } else if (returnVal == JFileChooser.CANCEL_OPTION) {
183:         selectionFinished = true;
184:     }
185:     setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
186: }
187: }
188: });
189:
190: /*
191: new Thread(new Runnable() {
192: public void run() {
193:     boolean selectionFinished = false;
194:     JFileChooser fc = new JFileChooser(new File(Configuration.root));
195:     fc.addChoosableFileFilter(new FileNameExtensionFilter("File ZIP", "zip"));
196:     FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
197:     fc.removeChoosableFileFilter(acceptAllFileFilter);
198:     while (!selectionFinished) {
199:         int returnVal = fc.showSaveDialog(SaveXMLFilePanel.this);
200:         if (returnVal == JFileChooser.APPROVE_OPTION) {
201:             File file = fc.getSelectedFile();
202:             if (file.exists()) {
203:                 int n = JOptionPane.showConfirmDialog(SaveXMLFilePanel.this, "Do you want to overwrite the selected file?",
204: "Warning", JOptionPane.YES_NO_OPTION);
205:                 if (n == 0) {
206:                     fileNameTextField.setText(file.getAbsolutePath());
207:                     selectionFinished = true;
208:                 }
209:             } else {
210:                 fileNameTextField.setText(file.getAbsolutePath());
211:                 selectionFinished = true;
212:             }
213:         } else if (returnVal == JFileChooser.CANCEL_OPTION) {
214:             selectionFinished = true;
215:         }
216:     }
217:     setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
218: }
```

```
218:     }).start();*/
219:
220:     }//GEN-LAST:event_browseButtonActionPerformed
221:
222:     private void closeButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_closeButtonActionPerformed
223:         try {
224:             saveData();
225:             rootPanel.addPanel(new ConfigurationManagerPanel(rootPanel));
226: } //GEN-LAST:event_closeButtonActionPerformed
227:         catch(Exception ex){
228:             JOptionPane.showMessageDialog(this, ex.getMessage(), "Error", JOptionPane.ERROR_MESSAGE);
229:             Logger.getLogger(SaveXMLFilePanel.class.getName()).log(Level.SEVERE, null, ex);
230:         }
231:     }
232:
233:     @Override
234:     public void saveData() throws Exception {
235:         bIROAdaptorConfiguration.setExportFilePath(fileNameTextField.getText());
236:
237:     }
238:     // Variables declaration - do not modify//GEN-BEGIN:variables
239:     private javax.swing.JButton browseButton;
240:     private javax.swing.JPanel buttonPanel;
241:     private javax.swing.JButton closeButton;
242:     private javax.swing.JTextField fileNameTextField;
243:     private javax.swing.JLabel jLabel1;
244:     private javax.swing.JButton previousButton;
245:     // End of variables declaration//GEN-END:variables
246: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      UnitOfMeasurementPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.adaptor;
31:
32: import eu.biro.birobox.panel.*;
33: import eu.biro.adaptor2.field.BIROField;
34: import eu.biro.adaptor2.field.NumericBIROField;
35: import eu.biro.adaptor2.unit.MeasurementUnit;
36: import java.util.Collection;
37: import javax.swing.DefaultComboBoxModel;
38:
39: public class UnitOfMeasurementPanel extends FieldPanel {
40:
41:     private javax.swing.JComboBox measurementUnitComboBox;
42:     private Collection<MeasurementUnit> units;
43:     private static UnitOfMeasurementPanel instance;
44:
45:     /** Creates new form UnitOfMeasurementPanel */
```

```
46: private UnitOfMeasurementPanel() {
47:     initComponents();
48:     measurementUnitComboBox = new javax.swing.JComboBox();
49:     add(measurementUnitComboBox);
50: }
51:
52: @Override
53: public void load(BIROField biroField) {
54:     NumericBIROField f = (NumericBIROField) biroField;
55:     Object o;
56:     measurementUnitComboBox.setModel(new DefaultComboBoxModel(f.getSupportedUnits().toArray()));
57:     if (f.getSelectedUnit() == null) {
58:         o = measurementUnitComboBox.getModel().getElementAt(0);
59:     } else {
60:         o = f.getSelectedUnit();
61:     }
62:     measurementUnitComboBox.setSelectedItem(o);
63: }
64:
65: @Override
66: public void save(BIROField biroField) {
67:     NumericBIROField f = (NumericBIROField) biroField;
68:     f.setSelectedUnit((MeasurementUnit) measurementUnitComboBox.getSelectedItem());
69: }
70:
71:
72: public static UnitOfMeasurementPanel getInstance() {
73:     if (instance == null) {
74:         instance = new UnitOfMeasurementPanel();
75:     }
76:     return instance;
77: }
78:
79: /** This method is called from within the constructor to
80:  * initialize the form.
81:  * WARNING: Do NOT modify this code. The content of this method is
82:  * always regenerated by the Form Editor.
83:  */
84: // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
85: private void initComponents() {
86:
87:     chooseUnitLabel = new javax.swing.JLabel();
88:
89:     setLayout(new java.awt.GridLayout(2, 1, 10, 10));
90:
```

```
91:         chooseUnitLabel.setText("Choose the unit of measurement:");
92:         add(chooseUnitLabel);
93:     }// </editor-fold>//GEN-END:initComponents
94:     // Variables declaration - do not modify//GEN-BEGIN:variables
95:     private javax.swing.JLabel chooseUnitLabel;
96:     // End of variables declaration//GEN-END:variables
97: }
98:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      CommunicationSoftwarePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * GPL Copyright, The BIRO Project
27:  *
28:  */
29: package eu.biro.birobox.panel.communicationSoftware;
30:
31: import eu.biro.birobox.panel.statisticalEngine.*;
32: import eu.biro.birobox.configuration.CommunicationSoftwareConfiguration;
33: import eu.biro.birobox.configuration.Configuration;
34: import eu.biro.birobox.models.StatisticalObjectTableCellRenderer;
35: import eu.biro.birobox.models.StatisticalObjectTableModel;
36: import eu.biro.birobox.panel.*;
37: import eu.biro.birobox.panel.ChildrenPanel;
38: import eu.biro.communication.common.BiroUpload;
39: import eu.biro.communication.common.CommunicationConfiguration;
40: import eu.biro.communication.common.UploadLoaderException;
41: import eu.biro.communication.impl.BiroServiceInvocation;
42: import java.awt.Cursor;
43: import java.awt.Toolkit;
44: import java.io.BufferedReader;
45: import java.io.InputStreamReader;
```

```
46: import java.io.File;
47: import java.io.FileInputStream;
48: import java.io.FileNotFoundException;
49: import java.io.FileOutputStream;
50: import java.io.FileReader;
51: import java.io.IOException;
52: import java.sql.Timestamp;
53: import java.text.ParseException;
54: import java.text.SimpleDateFormat;
55: import java.util.Date;
56: import java.util.Iterator;
57: import java.util.Vector;
58: import java.util.logging.Handler;
59: import java.util.logging.Level;
60: import java.util.logging.LogRecord;
61: import java.util.logging.Logger;
62: import java.util.regex.Matcher;
63: import java.util.regex.Pattern;
64: import java.util.zip.ZipEntry;
65: import java.util.zip.ZipOutputStream;
66: import javax.swing.InputVerifier;
67: import javax.swing.JComponent;
68: import javax.swing.JOptionPane;
69: import javax.swing.JTextField;
70: import javax.swing.event.ListSelectionEvent;
71: import javax.swing.event.ListSelectionListener;
72:
73: public class CommunicationSoftwarePanel extends ChildrenPanel {
74:
75:     private RootPanel rootPanel;
76:     private CommunicationSoftwareConfiguration communicationSoftwareConfiguration;
77:     private String selectedStatisticalObjectName;
78:     private StatisticalObject selectedStatisticalObject;
79:     private Vector<StatisticalObject> statisticalObjects;
80:
81:     /** Creates new form StatisticalEnginePanel */
82:     public CommunicationSoftwarePanel(RootPanel rootPanel) {
83:         this.rootPanel = rootPanel;
84:         communicationSoftwareConfiguration =
CommunicationSoftwareConfiguration.getCommunicationSoftwareConfiguration();
85:         initComponents();
86:         initIPTextFields();
87:         loadData();
88:         statisticalObjects =
(CommunicationSoftwareConfiguration.getCommunicationSoftwareConfiguration()).getStatisticalObjects();
```

```
89:
90:     StatisticalObjectTableModel dm = new StatisticalObjectTableModel();
91:     statisticalObjectsTable.setModel(dm);
92:     statisticalObjectsTable.getSelectionModel().addListSelectionListener(new
StatisticalObjectTableListSelectionListener());
93:     statisticalObjectsTable.setDefaultRenderer(StatisticalObject.class, new
StatisticalObjectTableCellRenderer()); //
94:
95:
96: }
97:
98: /** This method is called from within the constructor to
99:  * initialize the form.
100:  * WARNING: Do NOT modify this code. The content of this method is
101:  * always regenerated by the Form Editor.
102:  */
103: // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
104: private void initComponents() {
105:
106:     ipAddressTextField1 = new javax.swing.JTextField();
107:     dotLabel1 = new javax.swing.JLabel();
108:     ipAddressTextField2 = new javax.swing.JTextField();
109:     dotLabel3 = new javax.swing.JLabel();
110:     ipAddressTextField3 = new javax.swing.JTextField();
111:     jLabel3 = new javax.swing.JLabel();
112:     ipAddressTextField4 = new javax.swing.JTextField();
113:     label = new javax.swing.JLabel();
114:     buttonPanel = new javax.swing.JPanel();
115:     jPanel1 = new javax.swing.JPanel();
116:     jPanel2 = new javax.swing.JPanel();
117:     runCommunicationSoftwareButton = new javax.swing.JButton();
118:     jScrollPane1 = new javax.swing.JScrollPane();
119:     statisticalObjectsTable = new javax.swing.JTable();
120:     jLabel1 = new javax.swing.JLabel();
121:     centralSystemIPAddressLabel = new javax.swing.JLabel();
122:     webServiceEndpointTextField = new javax.swing.JTextField();
123:     dotLabel1.setText(".");
124:     dotLabel3.setText(".");
125:     jLabel3.setText(".");
126:     setBorder(javax.swing.BorderFactory.createTitledBorder("Communicator Configuration"));
127:     label.setText("Please choose a statistical object to be sent:");
128:     buttonPanel.setLayout(new java.awt.GridLayout(1, 0));
129:     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
130:     jPanel1.setLayout(jPanel1Layout);
131:     jPanel1Layout.setHorizontalGroup(
```

BIROBox/src/eu/biro/birobox/panel/communicationSoftware/CommunicationSoftwarePanel.java

```
132:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
133:         .addGap(0, 264, Short.MAX_VALUE)
134:     );
135:     jPanel1Layout.setVerticalGroup(
136:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
137:         .addGap(0, 23, Short.MAX_VALUE)
138:     );
139:
140:     buttonPanel.add(jPanel1);
141:
142:     runCommunicationSoftwareButton.setText("send");
143:     runCommunicationSoftwareButton.setEnabled(false);
144:     runCommunicationSoftwareButton.addActionListener(new java.awt.event.ActionListener() {
145:         public void actionPerformed(java.awt.event.ActionEvent evt) {
146:             runCommunicationSoftwareButtonActionPerformed(evt);
147:         }
148:     });
149:
150:     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
151:     jPanel2.setLayout(jPanel2Layout);
152:     jPanel2Layout.setHorizontalGroup(
153:         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
154:         .addComponent(runCommunicationSoftwareButton, javax.swing.GroupLayout.DEFAULT_SIZE, 264,
Short.MAX_VALUE)
155:     );
156:     jPanel2Layout.setVerticalGroup(
157:         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
158:         .addComponent(runCommunicationSoftwareButton)
159:     );
160:
161:     buttonPanel.add(jPanel2);
162:     jScrollPane1.setViewportView(statisticalObjectsTable);
163:     jLabel1.setText("Send the following statistical object:");
164:     centralSystemIPAddressLabel.setText("Central BIRO System IP address");
165:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
166:     this.setLayout(layout);
167:     layout.setHorizontalGroup(
168:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
169:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup()
170:             .addComponent(centralSystemIPAddressLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 162,
Short.MAX_VALUE)
171:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
172:             .addComponent(webServiceEndpointTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 362,
Short.MAX_VALUE)
173:             .addGroup(layout.createSequentialGroup()
```

```
174:         .addComponent(label)
175:         .addContainerGap()
176:         .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE, 528, Short.MAX_VALUE)
177:         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 528, Short.MAX_VALUE)
178:         .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 528, Short.MAX_VALUE)
179:     );
180:     layout.setVerticalGroup(
181:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
182:         .addGroup(layout.createSequentialGroup())
183:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
184:             .addComponent(centralSystemIPAddressLabel)
185:             .addComponent(webServiceEndpointTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
186:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
187:         .addComponent(label)
188:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
189:         .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 227,
javax.swing.GroupLayout.PREFERRED_SIZE)
190:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
191:         .addComponent(jLabel1)
192:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 49, Short.MAX_VALUE)
193:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
194:     );
195: } // </editor-fold> // GEN-END: initComponents
196: private void runCommunicationSoftwareButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_runCommunicationSoftwareButtonActionPerformed
197:     ZipOutputStream zipOutputStream = null;
198:     FileOutputStream outputStream = null;
199:     try {
200:
201:         saveData();
202:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
203:         File zipfile = new File(Configuration.root + "_se_/sending/" + selectedStatisticalObjectName + ".zip");
204:         outputStream = new FileOutputStream(zipfile);
205:         zipOutputStream = new ZipOutputStream(outputStream);
206:         zipDir(Configuration.root + "_se_/output/data/" + selectedStatisticalObjectName, zipOutputStream,
Configuration.root + "_se_/output/data/" + selectedStatisticalObjectName + "/");
207:         zipOutputStream.close();
208:         // invoke CommunicationSoftware sending
209:         String result = startCommunicationSoftware();
210:         if (result.equals("error...")) {
211:             throw new Exception("An error occurred during the transmission.\nPlease contact the support.");
212:         }
213:         //aggiorno la data di invio
```

BIROBox/src/eu/biro/birobox/panel/communicationSoftware/CommunicationSoftwarePanel.java

```
214:         Date d = new Date();
215:         selectedStatisticalObject.addSendingTimestamp(new Timestamp(d.getTime()));
216:         statisticalObjectsTable.repaint();
217:         JOptionPane.showMessageDialog(this, result, "Sending result", JOptionPane.PLAIN_MESSAGE);
218:
219:     } catch (Exception ex) {
220:         JOptionPane.showMessageDialog(this, ex.getMessage(), "ERROR", JOptionPane.ERROR_MESSAGE);
221:         Logger.getLogger(CommunicationSoftwarePanel.class.getName()).log(Level.SEVERE, null, ex);
222:     } finally {
223:         setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
224:     }
225:
226: } //GEN-LAST:event_runCommunicationSoftwareButtonActionPerformed
227:
228:     public void loadData() {
229:         try {
230:             communicationSoftwareConfiguration =
CommunicationSoftwareConfiguration.getCommunicationSoftwareConfiguration();
231:
232:             for (String s : new File(Configuration.root + "_se_/output/data").list()) {
233:                 if (!communicationSoftwareConfiguration.containsStatisticalObject(s)) {
234:                     communicationSoftwareConfiguration.getStatisticalObjects().add(new StatisticalObject(s));
235:                 }
236:             }
237:             Iterator<StatisticalObject> i = communicationSoftwareConfiguration.getStatisticalObjects().iterator();
238:             Vector<StatisticalObject> statisticalObjectsToBeRemoved = new Vector<StatisticalObject>();
239:
240:             while (i.hasNext()) {
241:                 StatisticalObject so = i.next();
242:                 String filename = so.getId();
243:                 if (!(new File(Configuration.root + "_se_/output/data/" + filename)).exists()) {
244:                     statisticalObjectsToBeRemoved.add(so);
245:                 }
246:             }
247:             i = statisticalObjectsToBeRemoved.iterator();
248:             while (i.hasNext()) {
249:                 StatisticalObject so = i.next();
250:                 communicationSoftwareConfiguration.getStatisticalObjects().remove(so);
251:             }
252:
253:             webServiceEndpointTextField.setText(communicationSoftwareConfiguration.getWebServiceEndpoint());
254:
255:         } catch (ParseException ex) {
256:             Logger.getLogger(CommunicationSoftwarePanel.class.getName()).log(Level.SEVERE, null, ex);
257:         }
```

```
258:
259:     }
260:
261:     @Override
262:     public void saveData() throws Exception {
263:         communicationSoftwareConfiguration.setWebServiceEndpoint(webServiceEndpointTextField.getText());
264:         communicationSoftwareConfiguration.saveToFile(new
File(Configuration.CommunicationSoftwareConfigurationPath));
265:
266:     }
267:
268:     public void initIPTextFields() {
269:         InputVerifier verifier = new InputVerifier() {
270:
271:             Pattern pat = Pattern.compile("\\b(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?");
272:
273:             @Override
274:             public boolean shouldYieldFocus(JComponent input) {
275:                 boolean inputOK = verify(input);
276:                 if (inputOK) {
277:                     return true;
278:                 } else {
279:                     Toolkit.getDefaultToolkit().beep();
280:                     return false;
281:                 }
282:             }
283:
284:             public boolean verify(JComponent input) {
285:                 JTextField field = (JTextField) input;
286:                 Matcher m = pat.matcher(field.getText());
287:                 return m.matches();
288:             }
289:         };
290:
291:         ipAddressTextField1.setInputVerifier(verifier);
292:         ipAddressTextField2.setInputVerifier(verifier);
293:         ipAddressTextField3.setInputVerifier(verifier);
294:         ipAddressTextField4.setInputVerifier(verifier);
295:     }
296:     // Variables declaration - do not modify//GEN-BEGIN:variables
297:     private javax.swing.JPanel buttonPanel;
298:     private javax.swing.JLabel centralSystemIPAddressLabel;
299:     private javax.swing.JLabel dotLabel1;
300:     private javax.swing.JLabel dotLabel3;
301:     private javax.swing.JTextField ipAddressTextField1;
```

```

302: private javax.swing.JTextField ipAddressTextField2;
303: private javax.swing.JTextField ipAddressTextField3;
304: private javax.swing.JTextField ipAddressTextField4;
305: private javax.swing.JLabel jLabel1;
306: private javax.swing.JLabel jLabel3;
307: private javax.swing.JPanel jPanel1;
308: private javax.swing.JPanel jPanel2;
309: private javax.swing.JScrollPane jScrollPane1;
310: private javax.swing.JLabel label;
311: private javax.swing.JButton runCommunicationSoftwareButton;
312: private javax.swing.JTable statisticalObjectsTable;
313: private javax.swing.JTextField webServiceEndpointTextField;
314: // End of variables declaration//GEN-END:variables
315:
316: private void zipDir(String dir2zip, ZipOutputStream zipOutputStream, String root) {
317:     FileInputStream fileInputStream = null;
318:     BufferedInputStream bufferedInputStream = null;
319:     File zipDir = null;
320:     int BUFFER = 2048;
321:     int count = 0;
322:     byte[] data = new byte[BUFFER];
323:
324:     try {
325:         zipDir = new File(dir2zip);
326:         File[] dirList = zipDir.listFiles();
327:
328:         //loop through dirList, and zip the files
329:         for (int i = 0; i < dirList.length; i++) {
330:             if (dirList[i].isDirectory()) {
331:                 //if the File object is a directory, call this
332:                 //function again to add its content recursively
333:                 String filePath = dirList[i].getPath();
334:                 zipDir(filePath, zipOutputStream, root);
335:                 //loop again
336:                 continue;
337:             }
338:             //if we reached here, the File object f was not a directory
339:             //create a FileInputStream on top of f
340:             fileInputStream = new FileInputStream(dirList[i]);
341:             bufferedInputStream = new BufferedInputStream(fileInputStream, BUFFER);
342:             ZipEntry anEntry = new ZipEntry(dirList[i].getPath().substring(root.length()));
343:             zipOutputStream.putNextEntry(anEntry);
344:             System.out.println("Adding: " + dirList[i].getPath().substring(root.length()));
345:             while ((count = fileInputStream.read(data)) != -1) {
346:                 zipOutputStream.write(data, 0, count);

```

```
347:         }
348:         bufferedInputStream.close();
349:
350:     }
351:     } catch (Exception e) {
352:         Logger.getLogger(CommunicationSoftwarePanel.class.getName()).log(Level.SEVERE, null, e);
353:     }
354: }
355:
356: private String startCommunicationSoftware() throws Exception {
357:     String result = "";
358:     saveData();
359:     String filename = Configuration.root + "_se_/sending/" + selectedStatisticalObjectName + ".zip";
360:     CommunicationConfiguration conf = new CommunicationConfiguration();
361:     conf.setClientConfiguration(Configuration.root + "_cs_/config/client_axis2_conf.xml");
362:     conf.setRepositoryPath(Configuration.root + "_cs_/axis2-1.4/repository");
363:     conf.setServiceEndpoint(communicationSoftwareConfiguration.getWebServiceEndpoint());
364:     BufferedReader inputStream = null;
365:     int i;
366:
367:     inputStream = new BufferedReader(new FileReader(new File(Configuration.root + "_se_/output/data/" +
selectedStatisticalObjectName + "/info.csv")));
368:     String l;
369:     while ((l = inputStream.readLine()) != null) {
370:
371:         if (l.startsWith("centre id: ")) {
372:             selectedStatisticalObject.setCentreID((l.substring(11)));
373:         } else if (l.startsWith("startdate: ")) {
374:             selectedStatisticalObject.setStartDate(new SimpleDateFormat("yyyy-MM-dd").parse(l.substring(11)));
375:         } else if (l.startsWith("enddate:")) {
376:             selectedStatisticalObject.setEndDate(new SimpleDateFormat("yyyy-MM-dd").parse(l.substring(9)));
377:         }
378:
379:     }
380:     BiroUpload upload = BiroUpload.Factory.newInstance(selectedStatisticalObject.getStartDate(),
selectedStatisticalObject.getEndDate(), selectedStatisticalObject.getCentreID(), filename);
381:     BiroServiceInvocation serviceInvocation = new BiroServiceInvocation();
382:
383:     result = serviceInvocation.invokeTransfer(upload, conf, false);
384:     return result;
385: }
386:
387: private class StatisticalObjectTableListSelectionListener implements ListSelectionListener {
388:
389:     public StatisticalObjectTableListSelectionListener() {
```

```
390:         super();
391:     }
392:
393:     public void valueChanged(ListSelectionEvent e) {
394:         if (!e.getValueIsAdjusting()) {
395:             //if (statisticalObjectsTable.getSelectedColumn() == 3) {
396:                 selectedStatisticalObject = (StatisticalObject)
statisticalObjects.elementAt(statisticalObjectsTable.getSelectedRow());
397:                 selectedStatisticalObjectName = selectedStatisticalObject.toString();
398:                 jLabel1.setText("Send the following statistical object: " + selectedStatisticalObjectName);
399:                 runCommunicationSoftwareButton.setEnabled(true);
400:             //}
401:             repaint();
402:         }
403:     }
404: }
405: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalObject.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.communicationSoftware;
31:
32: import java.io.Serializable;
33: import java.sql.Timestamp;
34: import java.text.ParseException;
35: import java.text.SimpleDateFormat;
36: import java.util.Date;
37: import java.util.Vector;
38:
39:
40: public class StatisticalObject implements Serializable {
41:
42:     private static final long serialVersionUID = -4873657210438921516L;
43:     private String id;
44:     private Timestamp creationTimestamp;
45:     private Vector<Timestamp> sendingTimestamps;
```

```
46: private String centreID;
47: private Date startDate;
48: private Date endDate;
49:
50: public StatisticalObject(String id) throws ParseException {
51:     this.id = id;
52:     this.sendingTimestamps = new Vector<Timestamp>();
53:     SimpleDateFormat formatter = new SimpleDateFormat("ddMMyyhhmmss");
54:     Date utilDate = formatter.parse(id.substring(1));
55:     this.creationTimestamp = new Timestamp(utilDate.getTime());
56: }
57:
58: public StatisticalObject(String id,String centreID, Date startDate, Date endDate) throws ParseException {
59:     this.id = id;
60:     this.sendingTimestamps = new Vector<Timestamp>();
61:     SimpleDateFormat formatter = new SimpleDateFormat("ddMMyyhhmmss");
62:     Date utilDate = formatter.parse(id.substring(1));
63:     this.creationTimestamp = new Timestamp(utilDate.getTime());
64:     this.endDate=endDate;
65:     this.startDate= startDate;
66:     this.centreID=centreID;
67: }
68:
69: public Timestamp getCreationTimestamp() {
70:     return creationTimestamp;
71: }
72:
73: public String getId() {
74:     return id;
75: }
76:
77: public Vector<Timestamp> getSendingTimestamps() {
78:     return sendingTimestamps;
79: }
80:
81: public void setSendingTimestamps(Vector<Timestamp> sendingTimestamps) {
82:     this.sendingTimestamps = sendingTimestamps;
83: }
84:
85: public void addSendingTimestamp(Timestamp sendingTimestamp) {
86:     sendingTimestamps.add(sendingTimestamp);
87: }
88:
89: public String getCentreID() {
90:     return centreID;
```

```
91:     }
92:
93:     public void setCentreID(String centreID) {
94:         this.centreID = centreID;
95:     }
96:
97:     public Date getEndDate() {
98:         return endDate;
99:     }
100:
101:     public void setEndDate(Date endDate) {
102:         this.endDate = endDate;
103:     }
104:
105:
106:     public Date getStartDate() {
107:         return startDate;
108:     }
109:
110:     public void setStartDate(Date startDate) {
111:         this.startDate = startDate;
112:     }
113:
114:     @Override
115:     public String toString() {
116:         return this.id;
117:     }
118: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      DatabaseConfigurationPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.databaseManager;
31:
32: import eu.biro.birobox.panel.*;
33: import eu.biro.birobox.configuration.BIROBoxConfiguration;
34: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
35: import eu.biro.adaptor2.DBMSDriver;
36: import eu.biro.adaptor2.BIROAdaptorConfiguration;
37: import eu.biro.birobox.configuration.BIROAdaptorConfigurationList;
38: import eu.biro.birobox.configuration.Configuration;
39: import eu.biro.birobox.models.DBMSDriverComboBoxModel;
40: import java.io.IOException;
41: import java.sql.SQLException;
42: import java.util.logging.Level;
43: import java.util.logging.Logger;
44: import javax.swing.DefaultComboBoxModel;
45: import javax.swing.JOptionPane;
```

```
46: import javax.swing.JTextField;
47: import org.hibernate.HibernateException;
48: import org.hibernate.dialect.Dialect;
49: import org.hibernate.dialect.DialectFactory;
50: import eu.biro.birobox.utils.ConnectionManager;
51: import eu.biro.birobox.utils.CustomDBMSDriver;
52: import java.io.File;
53:
54:
55: public class DatabaseConfigurationPanel extends ChildrenPanel {
56:
57:     private RootPanel rootPanel;
58:     BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
59:     BIROBoxConfiguration bIROBoxConfiguration;
60:     BIROAdaptorConfiguration bIROAdaptorConfiguration;
61:     BIROAdaptorConfigurationList bIROAdaptorConfigurationList;
62:
63:     /** Creates new form NewJPanel */
64:     public DatabaseConfigurationPanel(RootPanel rootPanel) throws IOException {
65:         this.rootPanel = rootPanel;
66:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
67:         bIROBoxConfiguration = BIROBoxConfiguration.getBIROBoxConfiguration();
68:         bIROAdaptorConfigurationList = bIROAdaptorConfigurationList.getBIROAdaptorConfigurationList();
69:         bIROAdaptorConfiguration = bIROAdaptorConfigurationList.getCurrentConfiguration();
70:         initComponents();
71:         loadData();
72:     }
73:
74:     /** This method is called from within the constructor to
75:      * initialize the form.
76:      * WARNING: Do NOT modify this code. The content of this method is
77:      * always regenerated by the Form Editor.
78:      */
79:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
80:     private void initComponents() {
81:         bindingGroup = new org.jdesktop.beansbinding.BindingGroup();
82:
83:         bIRODatabaseConfigurationPanel = new javax.swing.JPanel();
84:         dbmsLabel = new javax.swing.JLabel();
85:         dbmsComboBox = new javax.swing.JComboBox();
86:         databaseHostAndPortLabel = new javax.swing.JLabel();
87:         databaseHostAndPortTextField = new javax.swing.JTextField();
88:         databaseUsernameLabel = new javax.swing.JLabel();
89:         databaseUsernameTextField = new javax.swing.JTextField();
90:         databasePasswordLabel = new javax.swing.JLabel();
```

BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseConfigurationPanel.java

```
91:         databasePasswordField = new javax.swing.JPasswordField();
92:         buttonPanel = new javax.swing.JPanel();
93:         leftButtonPanel = new javax.swing.JPanel();
94:         rightButtonPanel = new javax.swing.JPanel();
95:         saveConfigurationButton = new javax.swing.JButton();
96:         descriptionLabel1 = new javax.swing.JLabel();
97:         databaseNameTextField = new javax.swing.JTextField();
98:         databaseNameLabel = new javax.swing.JLabel();
99:         addDriverButton = new javax.swing.JButton();
100:        removeDriverButton = new javax.swing.JButton();
101:
102:        setMaximumSize(new java.awt.Dimension(500, 400));
103:        setMinimumSize(new java.awt.Dimension(500, 400));
104:
105:        bIRODatabaseConfigurationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("BIRO Database
Configuration"));
106:        bIRODatabaseConfigurationPanel.setMaximumSize(new java.awt.Dimension(400, 400));
107:        bIRODatabaseConfigurationPanel.setMinimumSize(new java.awt.Dimension(400, 400));
108:
109:        dbmsLabel.setText("BIRO DBMS driver");
110:
111:        org.jdesktop.beansbinding.Binding binding =
org.jdesktop.beansbinding.Bindings.createAutoBinding(org.jdesktop.beansbinding.AutoBinding.UpdateStrategy.READ_WRITE,
dbmsComboBox, org.jdesktop.beansbinding.ObjectProperty.create(), dbmsLabel, org.jdesktop.beansbinding.BeanProperty.create(
"labelFor"));
112:        bindingGroup.addBinding(binding);
113:
114:        dbmsComboBox.setModel(new DBMSDriverComboBoxModel(bIROBoxConfiguration.getDBMSDriverVector()));
115:
116:        databaseHostAndPortLabel.setText("BIRO database host and port");
117:
118:        databaseUsernameLabel.setText("BIRO database username");
119:
120:        databasePasswordLabel.setText("BIRO database password");
121:
122:        buttonPanel.setLayout(new java.awt.GridLayout(1, 2));
123:
124:        javax.swing.GroupLayout leftButtonPanelLayout = new javax.swing.GroupLayout(leftButtonPanel);
125:        leftButtonPanel.setLayout(leftButtonPanelLayout);
126:        leftButtonPanelLayout.setHorizontalGroup(
127:            leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
128:                .addGap(0, 251, Short.MAX_VALUE)
129:        );
130:        leftButtonPanelLayout.setVerticalGroup(
131:            leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

```
132:         .addGap(0, 25, Short.MAX_VALUE)
133:     );
134:
135:     buttonPanel.add(leftButtonPanel);
136:
137:     saveConfigurationButton.setText("Save BIRO database configuration");
138:     saveConfigurationButton.addActionListener(new java.awt.event.ActionListener() {
139:         public void actionPerformed(java.awt.event.ActionEvent evt) {
140:             saveConfigurationButtonActionPerformed(evt);
141:         }
142:     });
143:
144:     javax.swing.GroupLayout rightButtonPanelLayout = new javax.swing.GroupLayout(rightButtonPanel);
145:     rightButtonPanel.setLayout(rightButtonPanelLayout);
146:     rightButtonPanelLayout.setHorizontalGroup(
147:         rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
148:             .addComponent(saveConfigurationButton, javax.swing.GroupLayout.DEFAULT_SIZE, 251, Short.MAX_VALUE)
149:     );
150:     rightButtonPanelLayout.setVerticalGroup(
151:         rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
152:             .addGroup(rightButtonPanelLayout.createSequentialGroup()
153:                 .addComponent(saveConfigurationButton, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
154:                 .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
155:     );
156:
157:     buttonPanel.add(rightButtonPanel);
158:
159:     descriptionLabel1.setText("Configure Java database connection to BIRO database");
160:
161:     databaseNameLabel.setText("BIRO database name");
162:
163:     addDriverButton.setText("+");
164:     addDriverButton.addActionListener(new java.awt.event.ActionListener() {
165:         public void actionPerformed(java.awt.event.ActionEvent evt) {
166:             addDriverButtonActionPerformed(evt);
167:         }
168:     });
169:
170:     removeDriverButton.setText("-");
171:     removeDriverButton.addActionListener(new java.awt.event.ActionListener() {
172:         public void actionPerformed(java.awt.event.ActionEvent evt) {
173:             removeDriverButtonActionPerformed(evt);
174:         }
175:     });
```

```
176:
177:         javax.swing.GroupLayout bIRODatabaseConfigurationPanelLayout = new
javax.swing.GroupLayout(bIRODatabaseConfigurationPanel);
178:         bIRODatabaseConfigurationPanel.setLayout(bIRODatabaseConfigurationPanelLayout);
179:         bIRODatabaseConfigurationPanelLayout.setHorizontalGroup(
180:             bIRODatabaseConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
181:             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
bIRODatabaseConfigurationPanelLayout.createSequentialGroup())
182:         .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
183:             .addComponent(descriptionLabel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 495, Short.MAX_VALUE)
184:             .addGroup(bIRODatabaseConfigurationPanelLayout.createSequentialGroup())
185:         .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
186:             .addComponent(dbmsLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
187:             .addComponent(databaseHostAndPortLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
188:             .addComponent(databaseNameLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
189:             .addComponent(databaseUsernameLabel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
190:             .addComponent(databasePasswordLabel, javax.swing.GroupLayout.DEFAULT_SIZE, 133,
Short.MAX_VALUE))
191:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
192:         .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
193:             .addGroup(bIRODatabaseConfigurationPanelLayout.createSequentialGroup())
194:             .addComponent(dbmsComboBox, 0, 248, Short.MAX_VALUE)
195:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
196:             .addComponent(addDriverButton, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
197:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
198:             .addComponent(removeDriverButton, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
javax.swing.GroupLayout.PREFERRED_SIZE)
199:             .addGap(2, 2, 2))
200:             .addComponent(databaseUsernameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 350,
Short.MAX_VALUE)
201:             .addGroup(bIRODatabaseConfigurationPanelLayout.createSequentialGroup())
202:             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
203:             .addComponent(databaseNameTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 350,
Short.MAX_VALUE))
204:             .addComponent(databasePasswordField, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 350, Short.MAX_VALUE)
```

```
205:                                .addGroup( javax.swing.GroupLayout.Alignment.TRAILING,
bIRODatabaseConfigurationPanelLayout.createSequentialGroup()
206:                                .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
207:                                .addComponent(databaseHostAndPortTextField, javax.swing.GroupLayout.DEFAULT_SIZE,
350, Short.MAX_VALUE))))
208:                                .addGap(8, 8, 8))
209:                                .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 503, Short.MAX_VALUE)
210:                                );
211:                                bIRODatabaseConfigurationPanelLayout.setVerticalGroup(
212:                                bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.LEADING)
213:                                .addGroup( javax.swing.GroupLayout.Alignment.TRAILING,
bIRODatabaseConfigurationPanelLayout.createSequentialGroup()
214:                                .addComponent(descriptionLabel1)
215:                                .addGap(18, 18, 18)
216:                                .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
217:                                .addComponent(removeDriverButton)
218:                                .addComponent(addDriverButton)
219:                                .addComponent(dbmsLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
220:                                .addComponent(dbmsComboBox, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
221:                                .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
222:                                .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
223:                                .addComponent(databaseHostAndPortLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
224:                                .addComponent(databaseHostAndPortTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
225:                                .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
226:                                .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
227:                                .addComponent(databaseNameTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
228:                                .addComponent(databaseNameLabel))
229:                                .addGap(6, 6, 6)
230:                                .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
231:                                .addComponent(databaseUsernameTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE)
232:                                .addComponent(databaseUsernameLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
233:                                .addPreferredGap( javax.swing.LayoutStyle.ComponentPlacement.RELATED)
234:                                .addGroup(bIRODatabaseConfigurationPanelLayout.createParallelGroup( javax.swing.GroupLayout.Alignment.BASELINE)
```

```
235:                .addComponent(databasePasswordField, javax.swing.GroupLayout.DEFAULT_SIZE, 24, Short.MAX_VALUE)
236:                .addComponent(databasePasswordLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 23,
javax.swing.GroupLayout.PREFERRED_SIZE))
237:                .addGap(265, 265, 265)
238:                .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 25,
javax.swing.GroupLayout.PREFERRED_SIZE))
239:            );
240:
241:            javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
242:            this.setLayout(layout);
243:            layout.setHorizontalGroup(
244:                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
245:                .addComponent(bIRODatabaseConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
246:            );
247:            layout.setVerticalGroup(
248:                layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
249:                .addComponent(bIRODatabaseConfigurationPanel, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
250:            );
251:
252:            bindingGroup.bind();
253:        } // </editor-fold> // GEN-END: initComponents
254:        private void saveConfigurationButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_saveConfigurationButtonActionPerformed
255:            try {
256:                saveData();
257:                checkConnection();
258:                bIRODatabaseManagerConfiguration.saveToFile(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
259:                JOptionPane.showMessageDialog(DatabaseConfigurationPanel.this, "The configuration has been saved
correctly", "Information", JOptionPane.INFORMATION_MESSAGE);
260:                rootPanel.addPanel(new HomePanel(rootPanel));
261:            } catch (Exception ex) {
262:                String m;
263:                Logger.getLogger(DatabaseConfigurationPanel.class.getName()).log(Level.SEVERE, null, ex);
264:                if (ex instanceof HibernateException) {
265:                    m = "I can't find an appropriate Hibernate Dialect for this database.\nPlease check database
Configuration";
266:                } else {
267:                    m = "I can't establish a connection to the database.\nPlease check JDBC configuration";
268:                }
269:
270:                JOptionPane.showMessageDialog(DatabaseConfigurationPanel.this, m, "Warning",
JOptionPane.WARNING_MESSAGE);
```

```
271:         }
272:     } //GEN-LAST:event_saveConfigurationButtonActionPerformed
273:
274:     private void addDriverButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_addDriverButtonActionPerformed
275:
276:         JTextField dBMSNameTextField = new JTextField();
277:         JTextField driverClassTextField = new JTextField();
278:         JTextField driverUrlPatternTextField = new JTextField();
279:         JTextField driverJarPathTextField = new JTextField();
280:         int i = JOptionPane.showConfirmDialog(rootPanel, new Object[]{"DBMS name:", dBMSNameTextField, "driver
class:", driverClassTextField, "driver url pattern:", driverUrlPatternTextField, "jar path:", driverJarPathTextField},
"New DBMS Driver", JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE, null);
281:         System.out.println(i);
282:         if (i == 0) {
283:             if (!dBMSNameTextField.getText().equals("") && !driverClassTextField.getText().equals("") &&
!driverUrlPatternTextField.getText().equals("") && !driverJarPathTextField.getText().equals("")) {
284:                 ((DBMSDriverComboBoxModel) dbmsComboBox.getModel()).addElement(new
CustomDBMSDriver(dBMSNameTextField.getText(), driverClassTextField.getText(), driverUrlPatternTextField.getText(),
driverJarPathTextField.getText()));
285:                 dbmsComboBox.setSelectedIndex(((DBMSDriverComboBoxModel) (dbmsComboBox.getModel())).getSize() - 1);
286:                 dbmsComboBox.repaint();
287:             }
288:
289:         }
290:
291:     } //GEN-LAST:event_addDriverButtonActionPerformed
292:
293:     private void removeDriverButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_removeDriverButtonActionPerformed
294:         DBMSDriver d = (DBMSDriver) dbmsComboBox.getSelectedItem();
295:         boolean stillInUse = false;
296:
297:         if ((bIROAdaptorConfigurationList.getDBMSDriverUsersNumber(d) > 0)) {
298:             stillInUse = true;
299:         }
300:         if (!stillInUse) {
301:             int i = JOptionPane.showConfirmDialog(rootPanel, "Do you really want to remove/n the selected driver?"
, "Remove", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null);
302:             if (i == 0) {
303:                 int selectedIndex = dbmsComboBox.getSelectedIndex();
304:                 ((DBMSDriverComboBoxModel) (dbmsComboBox.getModel())).removeElementAt(selectedIndex);
305:                 if (selectedIndex != 0) {
306:                     dbmsComboBox.setSelectedIndex(selectedIndex - 1);
307:                 }

```

```
308:         dbmsComboBox.repaint();
309:     }
310: } else {
311:     JOptionPane.showMessageDialog(rootPanel, "This driver cannot be removed because is still in use",
"Warning", JOptionPane.WARNING_MESSAGE);
312: }
313: } //GEN-LAST:event_removeDriverButtonActionPerformed
314:
315: private void loadData() throws IOException {
316:     DBMSDriver d = bIRODatabaseManagerConfiguration.getBIRODBMSDriver();
317:     if (d != null) {
318:         ((DBMSDriverComboBoxModel) dbmsComboBox.getModel()).setSelectedDriver(d);
319:         databaseHostAndPortTextField.setText(d.getHostAndPort());
320:         databaseUsernameTextField.setText(d.getUser());
321:         databaseNameTextField.setText(d.getDatabaseName());
322:         databasePasswordField.setText(d.getPwd());
323:     } else {
324:         dbmsComboBox.setSelectedIndex(0);
325:     }
326: }
327:
328: public void saveData() throws Exception {
329:     DBMSDriver driver = ((DBMSDriver) dbmsComboBox.getModel().getSelectedItem());
330:     DBMSDriver driverClone = (DBMSDriver) driver.clone();
331:     driverClone.setHostAndPort(databaseHostAndPortTextField.getText());
332:     driverClone.setDatabaseName(databaseNameTextField.getText());
333:     driverClone.setUser(databaseUsernameTextField.getText());
334:     driverClone.setPwd(String.valueOf(databasePasswordField.getPassword()));
335:     bIRODatabaseManagerConfiguration.setBIRODBMSDriver(driverClone);
336:     //bIROBoxConfiguration.setDBMSDriverVector(dbMSDriverVector);
337: }
338:
339: private void checkConnection() throws ClassNotFoundException, SQLException, Exception {
340:     ConnectionManager manager =
ConnectionManager.initializeManager(bIRODatabaseManagerConfiguration.getBIRODBMSDriver());
341:     manager.connect();
342:     Dialect dialect = (DialectFactory.determineDialect(manager.getMetaData().getDatabaseProductName(),
manager.getMetaData().getDatabaseMajorVersion()));
343:     System.out.println(dialect.toString());
344:     if (dialect != null) {
345:         ((DBMSDriver) dbmsComboBox.getModel().getSelectedItem()).setHibernateDialect(dialect.toString());
346:         (bIRODatabaseManagerConfiguration.getBIRODBMSDriver()).setHibernateDialect(dialect.toString());
347:     } else {
348:         throw new HibernateException("");
349:     }
```

```
350:     }
351:     // Variables declaration - do not modify//GEN-BEGIN:variables
352:     private javax.swing.JButton addDriverButton;
353:     private javax.swing.JPanel bIRODatabaseConfigurationPanel;
354:     private javax.swing.JPanel buttonPanel;
355:     private javax.swing.JLabel databaseHostAndPortLabel;
356:     private javax.swing.JTextField databaseHostAndPortTextField;
357:     private javax.swing.JLabel databaseNameLabel;
358:     private javax.swing.JTextField databaseNameTextField;
359:     private javax.swing.JPasswordField databasePasswordField;
360:     private javax.swing.JLabel databasePasswordLabel;
361:     private javax.swing.JLabel databaseUsernameLabel;
362:     private javax.swing.JTextField databaseUsernameTextField;
363:     private javax.swing.JComboBox dbmsComboBox;
364:     private javax.swing.JLabel dbmsLabel;
365:     private javax.swing.JLabel descriptionLabel1;
366:     private javax.swing.JPanel leftButtonPanel;
367:     private javax.swing.JButton removeDriverButton;
368:     private javax.swing.JPanel rightButtonPanel;
369:     private javax.swing.JButton saveConfigurationButton;
370:     private org.jdesktop.beansbinding.BindingGroup bindingGroup;
371:     // End of variables declaration//GEN-END:variables
372: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      DatabaseManagerPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.databaseManager;
31:
32: import eu.biro.birobox.panel.*;
33: import java.awt.Cursor;
34: import java.io.File;
35: import java.io.IOException;
36: import java.sql.BatchUpdateException;
37: import java.util.logging.Level;
38: import java.util.logging.Logger;
39: import javax.swing.JFileChooser;
40: import javax.swing.SwingUtilities;
41: import test.BIRODatabaseManager;
42: import eu.biro.birobox.panel.RootPanel;
43: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
44: import eu.biro.birobox.configuration.Configuration;
45: import javax.swing.filechooser.FileFilter;
```

```
46: import javax.swing.filechooser.FileNameExtensionFilter;
47: import mergecreator.MergeCreator;
48:
49:
50: public class DatabaseManagerPanel extends ChildrenPanel {
51:
52:     private RootPanel rootPanel;
53:     BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
54:
55:     /** Creates new form NewJPanel */
56:     public DatabaseManagerPanel(RootPanel rootPanel) throws IOException {
57:         this.rootPanel = rootPanel;
58:         initComponents();
59:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
60:         loadData();
61:     }
62:
63:     /** This method is called from within the constructor to
64:     * initialize the form.
65:     * WARNING: Do NOT modify this code. The content of this method is
66:     * always regenerated by the Form Editor.
67:     */
68:     // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
69:     private void initComponents() {
70:
71:         jdbcConfigurationPanel = new javax.swing.JPanel();
72:         descriptionLabel = new javax.swing.JLabel();
73:         buttonPanel = new javax.swing.JPanel();
74:         leftButtonPanel = new javax.swing.JPanel();
75:         rightButtonPanel = new javax.swing.JPanel();
76:         runDatabaseManagerButton = new javax.swing.JButton();
77:         inputFolderTextfield = new javax.swing.JTextField();
78:         browseButton = new javax.swing.JButton();
79:
80:         setMaximumSize(new java.awt.Dimension(500, 400));
81:         setMinimumSize(new java.awt.Dimension(500, 400));
82:
83:         jdbcConfigurationPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Database Manager Panel"));
84:         jdbcConfigurationPanel.setMaximumSize(new java.awt.Dimension(400, 400));
85:         jdbcConfigurationPanel.setMinimumSize(new java.awt.Dimension(400, 400));
86:
87:         descriptionLabel.setText("Configure the input folder containing XML files ");
88:
89:         buttonPanel.setLayout(new java.awt.GridLayout(1, 2));
90:
```

BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseManagerPanel.java

```
91:         javax.swing.GroupLayout leftButtonPanelLayout = new javax.swing.GroupLayout(leftButtonPanel);
92:         leftButtonPanel.setLayout(leftButtonPanelLayout);
93:         leftButtonPanelLayout.setHorizontalGroup(
94:             leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
95:                 .addGap(0, 242, Short.MAX_VALUE)
96:         );
97:         leftButtonPanelLayout.setVerticalGroup(
98:             leftButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
99:                 .addGap(0, 24, Short.MAX_VALUE)
100:         );
101:
102:         buttonPanel.add(leftButtonPanel);
103:
104:         runDatabaseManagerButton.setText("Run BIRO Database Manager");
105:         runDatabaseManagerButton.addActionListener(new java.awt.event.ActionListener() {
106:             public void actionPerformed(java.awt.event.ActionEvent evt) {
107:                 runDatabaseManagerButtonActionPerformed(evt);
108:             }
109:         });
110:
111:         javax.swing.GroupLayout rightButtonPanelLayout = new javax.swing.GroupLayout(rightButtonPanel);
112:         rightButtonPanel.setLayout(rightButtonPanelLayout);
113:         rightButtonPanelLayout.setHorizontalGroup(
114:             rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
115:                 .addComponent(runDatabaseManagerButton, javax.swing.GroupLayout.DEFAULT_SIZE, 242, Short.MAX_VALUE)
116:         );
117:         rightButtonPanelLayout.setVerticalGroup(
118:             rightButtonPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
119:                 .addComponent(runDatabaseManagerButton, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE)
120:         );
121:
122:         buttonPanel.add(rightButtonPanel);
123:
124:         browseButton.setText("Browse");
125:         browseButton.addActionListener(new java.awt.event.ActionListener() {
126:             public void actionPerformed(java.awt.event.ActionEvent evt) {
127:                 browseButtonActionPerformed(evt);
128:             }
129:         });
130:
131:         javax.swing.GroupLayout jdbcConfigurationPanelLayout = new javax.swing.GroupLayout(jdbcConfigurationPanel);
132:         jdbcConfigurationPanel.setLayout(jdbcConfigurationPanelLayout);
133:         jdbcConfigurationPanelLayout.setHorizontalGroup(
134:             jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
```

BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseManagerPanel.java

```
135:         .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 0, Short.MAX_VALUE)
136:         .addGroup(jdbcConfigurationPanelLayout.createSequentialGroup())
137:         .addComponent(descriptionLabel)
138:         .addContainerGap()
139:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jdbcConfigurationPanelLayout.createSequentialGroup())
140:         .addComponent(inputFolderTextfield, javax.swing.GroupLayout.DEFAULT_SIZE, 389, Short.MAX_VALUE)
141:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
142:         .addComponent(browseButton, javax.swing.GroupLayout.PREFERRED_SIZE, 89,
javax.swing.GroupLayout.PREFERRED_SIZE))
143:     );
144:     jdbcConfigurationPanelLayout.setVerticalGroup(
145:         jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
146:         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jdbcConfigurationPanelLayout.createSequentialGroup())
147:         .addContainerGap()
148:         .addComponent(descriptionLabel)
149:         .addGap(31, 31, 31)
150:     .addGroup(jdbcConfigurationPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
151:         .addComponent(browseButton)
152:         .addComponent(inputFolderTextfield, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
153:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 404, Short.MAX_VALUE)
154:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE, 24,
javax.swing.GroupLayout.PREFERRED_SIZE))
155:     );
156:
157:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
158:     this.setLayout(layout);
159:     layout.setHorizontalGroup(
160:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
161:         .addComponent(jdbcConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
162:     );
163:     layout.setVerticalGroup(
164:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
165:         .addComponent(jdbcConfigurationPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
166:     );
167:     } // </editor-fold> // GEN-END: initComponents
168:     private void runDatabaseManagerButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_runDatabaseManagerButtonActionPerformed
169:
170:         try {
```

```
171:         saveData();
172:         bIRODatabaseManagerConfiguration.saveToFile(new
File(Configuration.bIRODatabaseManagerConfigurationFilePath));
173:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
174:         new Thread(new Runnable() {
175:
176:             public void run() {
177:                 String inputDirectory = bIRODatabaseManagerConfiguration.getDatabaseManagerInputDirectory();
178:                 String classDriver = bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getClassName();
179:                 String hibernateDialect =
bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getHibernateDialect();
180:                 String url = bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getUrl();
181:                 String username = bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getUser();
182:                 String password = bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getPwd();
183:                 BIRODatabaseManager bIRODatabaseManager = new BIRODatabaseManager(inputDirectory, classDriver,
hibernateDialect, url, username, password);
184:                 MergeCreator mergeCreator = new
MergeCreator(bIRODatabaseManagerConfiguration.getBIRODBMSDriver());
185:                 DatabaseManagerProgressPanel panel =
DatabaseManagerProgressPanel.createAndShowDatabaseManagerProgress(bIRODatabaseManager);
186:                 try {
187:                     panel.showText("Database Manager process is running. Please wait...");
188:                     bIRODatabaseManager.unmarshallAndStoreDirectory();
189:                     panel.showText("Database Manager is completing the BIRO local database. Please wait...");
190:                 } catch (Exception ex) {
191:                     Logger.getLogger(DatabaseManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
192:                     panel.showText("An error occurred during Database Manager process for::\n" +
ex.toString());
193:                     if (ex instanceof BatchUpdateException) {
194:                         ((BatchUpdateException) ex).getNextException().printStackTrace();
195:                     }
196:                 }
197:                 try {
198:                     mergeCreator.createMerge();
199:                     panel.showText("Database Manager process completed successfully");
200:                 } catch (Exception e) {
201:                     Logger.getLogger(DatabaseManagerPanel.class.getName()).log(Level.SEVERE, null, e);
202:                     panel.showText("An error occurred during Database Manager process for::\n" + e.toString());
203:                 } finally {
204:                     setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
205:                 }
206:             }
207:         }).start();
208:     } catch (Exception ex) {
209:         Logger.getLogger(DatabaseManagerPanel.class.getName()).log(Level.SEVERE, null, ex);
```

```
210:         }
211:     } //GEN-LAST:event_runDatabaseManagerButtonActionPerformed
212:
213:     private void browseButtonActionPerformed(java.awt.event.ActionEvent evt)
214: { //GEN-FIRST:event_browseButtonActionPerformed
215:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
216:         SwingUtilities.invokeLater(new Runnable() {
217:             public void run() {
218:                 JFileChooser fc = new JFileChooser(new File(Configuration.root +
219: Configuration.bIROAdaptorDefaultOutputFolder));
220:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("File ZIP", "zip"));
221:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
222:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
223:                 fc.setSelectionMode(JFileChooser.FILES_ONLY);
224:                 fc.setSelectedFile(new File(bIRODatabaseManagerConfiguration.getDatabaseManagerInputDirectory()));
225:                 int returnVal = fc.showOpenDialog(DatabaseManagerPanel.this);
226:                 if (returnVal == JFileChooser.APPROVE_OPTION) {
227:                     File file = fc.getSelectedFile();
228:                     inputFolderTextfield.setText(file.getAbsolutePath());
229:                     runDatabaseManagerButton.setEnabled(true);
230:                 }
231:                 setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
232:             }
233:         });
234:     } //GEN-LAST:event_browseButtonActionPerformed
235:
236:     private void loadData() throws IOException {
237:         inputFolderTextfield.setText(bIRODatabaseManagerConfiguration.getDatabaseManagerInputDirectory());
238:         if ((inputFolderTextfield.getText().equals("") | (inputFolderTextfield.getText() == null)) {
239:             runDatabaseManagerButton.setEnabled(false);
240:         } else {
241:             runDatabaseManagerButton.setEnabled(true);
242:         }
243:     }
244:
245:     public void saveData() throws IOException {
246:         bIRODatabaseManagerConfiguration.setDatabaseManagerInputDirectory(inputFolderTextfield.getText());
247:     }
248:
249:     // Variables declaration - do not modify //GEN-BEGIN:variables
250:     private javax.swing.JButton browseButton;
251:     private javax.swing.JPanel buttonPanel;
252:     private javax.swing.JLabel descriptionLabel;
```

```
253:     private javax.swing.JTextField inputFolderTextfield;
254:     private javax.swing.JPanel jdbcConfigurationPanel;
255:     private javax.swing.JPanel leftButtonPanel;
256:     private javax.swing.JPanel rightButtonPanel;
257:     private javax.swing.JButton runDatabaseManagerButton;
258:     // End of variables declaration//GEN-END:variables
259: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      DatabaseManagerProgressPanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.panel.databaseManager;
31:
32: import javax.swing.JComponent;
33: import javax.swing.JFrame;
34: import javax.swing.SwingUtilities;
35: import test.BIRODatabaseManager;
36: import util.DatabaseManagerProgressListener;
37:
38:
39: public class DatabaseManagerProgressPanel extends javax.swing.JPanel implements DatabaseManagerProgressListener {
40:
41:     /** Creates new form DatabaseManagerProgressPanel */
42:     public DatabaseManagerProgressPanel() {
43:         initComponents();
44:     }
45:
```

```

46:  /** This method is called from within the constructor to
47:   * initialize the form.
48:   * WARNING: Do NOT modify this code. The content of this method is
49:   * always regenerated by the Form Editor.
50:   */
51:  // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
52:  private void initComponents() {
53:
54:      progressBar = new javax.swing.JProgressBar();
55:      statusLabel = new javax.swing.JLabel();
56:      jScrollPane1 = new javax.swing.JScrollPane();
57:      outputTextArea = new javax.swing.JTextArea();
58:
59:      statusLabel.setText("BIRO Database Manager progress status: 0% completed");
60:
61:      outputTextArea.setColumns(20);
62:      outputTextArea.setEditable(false);
63:      outputTextArea.setRows(5);
64:      jScrollPane1.setViewportView(outputTextArea);
65:
66:      javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
67:      this.setLayout(layout);
68:      layout.setHorizontalGroup(
69:          layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
70:              .addGroup(layout.createSequentialGroup()
71:                  .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
72:                      .addComponent(jScrollPane1, javax.swing.GroupLayout.Alignment.LEADING,
73:                          javax.swing.GroupLayout.DEFAULT_SIZE, 366, Short.MAX_VALUE)
74:                      .addGroup(layout.createSequentialGroup()
75:                          .addComponent(statusLabel, javax.swing.GroupLayout.Alignment.LEADING)
76:                          .addComponent(progressBar, javax.swing.GroupLayout.DEFAULT_SIZE, 366, Short.MAX_VALUE)
77:                          .addGap(0))
78:                  )
79:              )
80:              .addGroup(layout.createSequentialGroup()
81:                  .addComponent(statusLabel)
82:                  .addGap(0)
83:                  .addComponent(progressBar, javax.swing.GroupLayout.PREFERRED_SIZE,
84:                      javax.swing.GroupLayout.PREFERRED_SIZE)
85:                  .addGap(0)
86:                  .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 162, Short.MAX_VALUE)
87:                  .addGap(0))
88:          );

```

```
89:    }// </editor-fold>//GEN-END:initComponents
90:    // Variables declaration - do not modify//GEN-BEGIN:variables
91:    private javax.swing.JScrollPane jScrollPane1;
92:    private javax.swing.JTextArea outputTextArea;
93:    private javax.swing.JProgressBar progressBar;
94:    private javax.swing.JLabel statusLabel;
95:    // End of variables declaration//GEN-END:variables
96:
97:    public void setMaximumForProgress(final int max) {
98:        SwingUtilities.invokeLater(new Runnable() {
99:
100:            public void run() {
101:                progressBar.setMaximum(max);
102:            }
103:        });
104:
105:    }
106:
107:    public void showText(String txt) {
108:        final String t = txt;
109:        SwingUtilities.invokeLater(new Runnable() {
110:
111:            public void run() {
112:                outputTextArea.append(t + "\n");
113:            }
114:        });
115:    }
116:
117:    public void updateProgressValue() {
118:        SwingUtilities.invokeLater(new Runnable() {
119:            public void run() {
120:                progressBar.setValue(progressBar.getValue() + 1);
121:                statusLabel.setText(String.format("Database Manager progress status: %d%% completed",
122:                ((progressBar.getValue() * 100) / progressBar.getMaximum())));
123:            }
124:        });
125:    }
126:    /**
127:     * Create the GUI and show it. As with all GUI code, this must run
128:     * on the event-dispatching thread.
129:     */
130:    public static DatabaseManagerProgressPanel createAndShowDatabaseManagerProgress(final BIRODatabaseManager dm) {
131:
132:        //Create and set up the content pane.
```

```
133:     final DatabaseManagerProgressPanel progressPanel = new DatabaseManagerProgressPanel();
134:     javax.swing.SwingUtilities.invokeLater(new Runnable() {
135:
136:         public void run() {
137:             //Create and set up the window.
138:             JFrame frame = new JFrame("Progress Status");
139:             frame.setResizable(false);
140:
141:
142:             JComponent newContentPane = progressPanel;
143:             newContentPane.setOpaque(true); //content panes must be opaque
144:             frame.setContentPane(newContentPane);
145:             frame.setLocationRelativeTo(null);
146:             dm.addProgressListener(progressPanel);
147:
148:             //Display the window.
149:             frame.pack();
150:             frame.setVisible(true);
151:         }
152:     });
153:     return progressPanel;
154:
155:
156: }
157: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      MyEngine.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.panel.statisticalEngine;
32:
33: import org.rosuda.JRI.RMainLoopCallbacks;
34: import org.rosuda.JRI.Rengine;
35:
36: public class MyEngine extends Rengine {
37:     private static MyEngine engine;
38:
39:     private MyEngine() {
40:     }
41:
42:     private MyEngine(String[] arg0, boolean arg1, RMainLoopCallbacks arg2) {
43:         super(arg0, arg1, arg2);
44:         //Rengine.DEBUG = 1;
45:     }
```

```
46:
47: public static MyEngine getInstance(){
48:     if (engine==null){
49:         String[] args = new String[]{"--vanilla"};
50:         engine = new MyEngine(args, false, null);
51:     }
52:     return (MyEngine) engine;
53: }
54:
55: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      ROutputFrame.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * GPL Copyright, The BIRO Project
27:  *
28:  */
29: package eu.biro.birobox.panel.statisticalEngine;
30:
31: import eu.biro.adaptor2.ProgressListener;
32: import eu.biro.birobox.configuration.Configuration;
33: import eu.biro.birobox.utils.FileListener;
34: import eu.biro.birobox.utils.FileMonitor;
35: import java.awt.Dimension;
36: import java.awt.Rectangle;
37: import java.awt.event.WindowAdapter;
38: import java.awt.event.WindowEvent;
39: import java.io.BufferedReader;
40: import java.io.ByteArrayOutputStream;
41: import java.io.File;
42: import java.io.FileReader;
43: import java.io.FileWriter;
44: import java.io.FilterOutputStream;
45: import java.io.IOException;
```

```
46: import java.io.InputStreamReader;
47: import java.io.OutputStream;
48: import java.io.PrintStream;
49: import java.util.logging.Level;
50: import java.util.logging.Logger;
51: import javax.swing.JFrame;
52: import javax.swing.SwingUtilities;
53: import org.rosuda.JRI.RMainLoopCallbacks;
54: import org.rosuda.JRI.Rengine;
55:
56: public class ROutputFrame extends javax.swing.JFrame implements RMainLoopCallbacks, ProgressListener, FileListener
{
57:
58:     PrintStream aPrintStream =
59:         new PrintStream(
60:             new FilteredStream(
61:                 new ByteArrayOutputStream()));
62:     boolean logFile;
63:     int counter;
64:     int prog;
65:     FileMonitor monitor;
66:
67:     /** Creates new form ROutputFrame */
68:     public ROutputFrame(boolean logFile) {
69:         initComponents();
70:         this.logFile = logFile;
71:         this.counter = 0;
72:         this.prog = 0;
73:         System.setOut(aPrintStream);
74:         System.setErr(aPrintStream);
75:
76:         setTitle("Statistical Engine Status");
77:         displayLog();
78:         statisticalEngineProgressBar.setMaximum(100);
79:         monitor = new FileMonitor(1000);
80:         //monitor.addFile(new File(Configuration.root + "_se\\output\\reports\\StatisticalEngine.log"));
81:         monitor.addFile(new File(Configuration.root + "_se\\statisticalEngineSinkFile.txt"));
82:
83:         monitor.addListener(this);
84:
85:         setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
86:         addWindowListener(new WindowAdapter() {
87:
88:             @Override
89:             public void windowClosing(WindowEvent e) {
```

```
90:         System.setErr(System.err);
91:         System.setOut(System.out);
92:         ((ROutputFrame) e.getSource()).close();
93:         ((ROutputFrame) e.getSource()).dispose();
94:
95:     }
96: });
97: }
98:
99: public void rWriteConsole(Rengine re, final String text, int arg2) {
100:     SwingUtilities.invokeLater(new Runnable() {
101:
102:         public void run() {
103:
104:             textArea.append(text);
105:
106:         }
107:     });
108: }
109:
110:
111: public void rBusy(Rengine re, final int which) {
112:     SwingUtilities.invokeLater(new Runnable() {
113:
114:         public void run() {
115:
116:             textArea.append("rBusy(" + which + ")");
117:
118:         }
119:     });
120: }
121:
122:
123: public String rReadConsole(Rengine re, String prompt, int addToHistory) {
124:
125:     textArea.append(prompt);
126:     try {
127:         BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
128:         String s = br.readLine();
129:         return (s == null || s.length() == 0) ? s : s + "\n";
130:     } catch (Exception e) {
131:         textArea.append("jriReadConsole exception: " + e.getMessage());
132:     }
133:
134:     return null;
```

```
135:     }
136:
137:     public void rShowMessage(Rengine re, String message) {
138:         textArea.append("rShowMessage \"" + message + "\"");
139:     }
140:
141:     public String rChooseFile(Rengine re, int newFile) {
142:         /*FileDialog fd = new FileDialog(f, (newFile == 0) ? "Select a file" : "Select a new file", (newFile == 0)
? FileDialog.LOAD : FileDialog.SAVE);
143:         fd.show();
144:         String res = null;
145:         if (fd.getDirectory() != null) {
146:             res = fd.getDirectory();
147:         }
148:         if (fd.getFile() != null) {
149:             res = (res == null) ? fd.getFile() : (res + fd.getFile());
150:         }
151:         return res;*/
152:         return "rChooseFile method TO DO";
153:     }
154:
155:     public void rFlushConsole(Rengine re) {
156:     }
157:
158:     public void rLoadHistory(Rengine re, String filename) {
159:     }
160:
161:     public void rSaveHistory(Rengine re, String filename) {
162:     }
163:
164:     class FilteredStream extends FilterOutputStream {
165:
166:         public FilteredStream(OutputStream aStream) {
167:             super(aStream);
168:         }
169:
170:         public void write(byte b[]) throws IOException {
171:             String aString = new String(b);
172:             textArea.append(aString);
173:         }
174:
175:         public void write(byte b[], int off, int len) throws IOException {
176:             String aString = new String(b, off, len);
177:
178:             textArea.append(aString);
```

```
179:         if (logFile) {
180:             FileWriter aWriter = new FileWriter("errorLogVale.log", true);
181:             aWriter.write(aString);
182:             aWriter.close();
183:         }
184:     }
185: }
186:
187:
188: public void displayLog() {
189:     Dimension dim = getToolkit().getScreenSize();
190:     Rectangle abounds = getBounds();
191:     Dimension dd = getSize();
192:     setLocation((dim.width - abounds.width) / 2,
193:                 (dim.height - abounds.height) / 2);
194:     setVisible(true);
195:     requestFocus();
196: }
197:
198: public void close() {
199:     this.aPrintStream.close();
200:     monitor.stop();
201:     monitor.removeListener(this);
202:     monitor.removeFile(new File(Configuration.root + "_se_\\statisticalEngineSinkFile.txt"));
203:     System.setOut(System.out);
204:     System.setErr(System.err);
205:
206:
207: }
208:
209: /** This method is called from within the constructor to
210:  * initialize the form.
211:  * WARNING: Do NOT modify this code. The content of this method is
212:  * always regenerated by the Form Editor.
213:  */
214: // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
215: private void initComponents() {
216:
217:     jPanel1 = new javax.swing.JPanel();
218:     statisticalEngineStatusLabel = new javax.swing.JLabel();
219:     jScrollPane1 = new javax.swing.JScrollPane();
220:     textArea = new javax.swing.JTextArea();
221:     statisticalEngineProgressBar = new javax.swing.JProgressBar();
222:
223:     setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
```

```
224:
225:     statisticalEngineStatusLabel.setText("Statistical Engine progress status:");
226:
227:     textArea.setColumns(20);
228:     textArea.setLineWrap(true);
229:     textArea.setRows(5);
230:     textArea.setWrapStyleWord(true);
231:     jScrollPane1.setViewportView(textArea);
232:
233:     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
234:     jPanel1.setLayout(jPanel1Layout);
235:     jPanel1Layout.setHorizontalGroup(
236:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
237:             .addGroup(jPanel1Layout.createSequentialGroup()
238:                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
239:                     .addGroup(jPanel1Layout.createSequentialGroup()
240:                         .addGap(10, 10, 10)
241:                         .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 319, Short.MAX_VALUE))
242:                     .addGroup(jPanel1Layout.createSequentialGroup()
243:                         .addContainerGap()
244:                         .addComponent(statisticalEngineStatusLabel))
245:                     .addGroup(jPanel1Layout.createSequentialGroup()
246:                         .addContainerGap()
247:                         .addComponent(statisticalEngineProgressBar, javax.swing.GroupLayout.DEFAULT_SIZE, 319,
Short.MAX_VALUE)))
248:                 .addContainerGap())
249:             );
250:     jPanel1Layout.setVerticalGroup(
251:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
252:             .addGroup(jPanel1Layout.createSequentialGroup()
253:                 .addContainerGap()
254:                 .addComponent(statisticalEngineStatusLabel)
255:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
256:                 .addComponent(statisticalEngineProgressBar, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
257:                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
258:                 .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE, 102, Short.MAX_VALUE)
259:                 .addContainerGap())
260:             );
261:
262:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
263:     getContentPane().setLayout(layout);
264:     layout.setHorizontalGroup(
265:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
266:             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
```

```
Short.MAX_VALUE)
267:         );
268:         layout.setVerticalGroup(
269:             layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
270:             .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
271:         );
272:
273:         pack();
274:     } // </editor-fold> // GEN-END: initComponents
275:     public static void CreateAndShowFrame() {
276:         // final BIROStatisticalEngine statisticalEngine = BIROStatisticalEngine.getBIROStatisticalEngine();
277:
278:         SwingUtilities.invokeLater(new Runnable() {
279:
280:             //private Object BIROStatisticalEngine;
281:             public void run() {
282:                 ROutputFrame r = new ROutputFrame(true);
283:                 //statisticalEngine.addProgressListener(r);
284:                 //FileMonitor monitor = new FileMonitor(10000);
285:                 // Add some files to listen for
286:                 //monitor.addFile(new File(Configuration.root + "_se_\\output\\reports\\StatisticalEngine.log"));
287:                 // Add a dummy listener
288:                 //monitor.addListener(r);
289:                 r.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
290:                 r.addWindowListener(new WindowAdapter() {
291:
292:                     @Override
293:                     public void windowClosing(WindowEvent e) {
294:                         //super.windowClosing(e);
295:
296:                         ((ROutputFrame) e.getSource()).close();
297:                         ((ROutputFrame) e.getSource()).dispose();
298:
299:                     }
300:                 });
301:
302:             }
303:         });
304:
305:     }
306:
307:     public void setMaximumForProgress(int max) {
308:         throw new UnsupportedOperationException("Not supported yet.");
309:     }
```

```
310:
311:     public void showText(String txt) {
312:         throw new UnsupportedOperationException("Not supported yet.");
313:     }
314:
315:     public void updateProgressValue() {
316:         throw new UnsupportedOperationException("Not supported yet.");
317:     }
318:
319:     public void fileChanged(File file) throws IOException {
320:         BufferedReader inputStream = null;
321:         int i;
322:         try {
323:             inputStream = new BufferedReader(new FileReader(file));
324:             String l;
325:             for (i = 0; i < counter; i++) {
326:                 l = inputStream.readLine();
327:             }
328:             textArea.append("\n");
329:             while ((l = inputStream.readLine()) != null) {
330:                 textArea.append(l + "\n");
331:                 if (l.startsWith("Indicator Number: ")) {
332:                     statisticalEngineProgressBar.setMaximum(Integer.parseInt(l.substring(18)));
333:                 } else if (l.startsWith("Indicator:") && l.endsWith("Done")) {
334:                     // textArea.append(l + "\n");
335:                     prog = prog + 1;
336:                     statisticalEngineProgressBar.setValue(prog);
337:                     statisticalEngineStatusLabel.setText(String.format("Statistical Engine progress status: %d%%
completed", ((statisticalEngineProgressBar.getValue() * 100) / statisticalEngineProgressBar.getMaximum())));
338:                 }
339:                 counter++;
340:             }
341:
342:         } catch (IOException ex) {
343:             Logger.getLogger(ROutputFrame.class.getName()).log(Level.SEVERE, null, ex);
344:         } finally {
345:             if (inputStream != null) {
346:                 inputStream.close();
347:             }
348:         }
349:
350:     }
351:     // Variables declaration - do not modify//GEN-BEGIN:variables
352:     private javax.swing.JPanel jPanel1;
353:     private javax.swing.JScrollPane jScrollPane1;
```

```
354:     private javax.swing.JProgressBar statisticalEngineProgressBar;  
355:     private javax.swing.JLabel statisticalEngineStatusLabel;  
356:     private javax.swing.JTextArea textArea;  
357:     // End of variables declaration//GEN-END:variables  
358: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * GPL Copyright, The BIRO Project
27:  *
28:  */
29: package eu.biro.birobox.panel.statisticalEngine;
30:
31: import eu.biro.birobox.configuration.StatisticalEngineConfiguration;
32: import eu.biro.birobox.configuration.Configuration;
33: import eu.biro.birobox.configuration.BIRODatabaseManagerConfiguration;
34: import eu.biro.birobox.configuration.CommunicationSoftwareConfiguration;
35: import eu.biro.birobox.panel.*;
36: import eu.biro.birobox.panel.ChildrenPanel;
37: import eu.biro.birobox.panel.communicationSoftware.StatisticalObject;
38: import eu.biro.birobox.utils.ConnectionManager;
39: import eu.biro.birobox.utils.JarFinder;
40: import java.awt.Cursor;
41: import java.io.File;
42: import java.io.FileWriter;
43: import java.text.SimpleDateFormat;
44: import java.util.Calendar;
45: import java.util.logging.Level;
```

```
46: import java.util.logging.Level;
47: import java.util.logging.Logger;
48: import javax.swing.JFileChooser;
49: import javax.swing.JOptionPane;
50: import javax.swing.JSpinner;
51: import javax.swing.SpinnerNumberModel;
52: import javax.swing.SwingUtilities;
53: import javax.swing.filechooser.FileFilter;
54: import javax.swing.filechooser.FileNameExtensionFilter;
55: import org.rosuda.JRI.Rengine;
56:
57: public class StatisticalEnginePanel extends ChildrenPanel {
58:
59:     private RootPanel rootPanel;
60:     private Calendar calendar;
61:     private int currentYear;
62:     private int duration;
63:     private StatisticalEngineConfiguration statisticalEngineConfiguration;
64:     private BIRODatabaseManagerConfiguration bIRODatabaseManagerConfiguration;
65:     private CommunicationSoftwareConfiguration communicationSoftwareConfiguration;
66:     boolean runningEnabled = false;
67:     boolean activitytablePresent = false;
68:
69:     /** Creates new form StatisticalEnginePanel */
70:     public StatisticalEnginePanel(RootPanel rootPanel) {
71:         this.rootPanel = rootPanel;
72:         calendar = Calendar.getInstance();
73:         currentYear = calendar.get(Calendar.YEAR);
74:         duration = 1;
75:         statisticalEngineConfiguration = StatisticalEngineConfiguration.getStatisticalEngineConfiguration();
76:         bIRODatabaseManagerConfiguration = BIRODatabaseManagerConfiguration.getBIRODatabaseManagerConfiguration();
77:         communicationSoftwareConfiguration =
CommunicationSoftwareConfiguration.getCommunicationSoftwareConfiguration();
78:         initComponents();
79:         SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
80:         loadData();
81:
82:
83:
84:     }
85:
86:     /** This method is called from within the constructor to
87:     * initialize the form.
88:     * WARNING: Do NOT modify this code. The content of this method is
89:     * always regenerated by the Form Editor.
```

```
90:      */
91:      // <editor-fold defaultstate="collapsed" desc="Generated Code">//GEN-BEGIN: initComponents
92:      private void initComponents() {
93:
94:          sourceButtonGroup = new javax.swing.ButtonGroup();
95:          centreIDLabel = new javax.swing.JLabel();
96:          yearNowLabel = new javax.swing.JLabel();
97:          startDateLabel = new javax.swing.JLabel();
98:          refDateLabel = new javax.swing.JLabel();
99:          yearNowSpinner = new javax.swing.JSpinner();
100:         buttonPanel = new javax.swing.JPanel();
101:         jPanel1 = new javax.swing.JPanel();
102:         jPanel2 = new javax.swing.JPanel();
103:         runStatisticalEngineButton = new javax.swing.JButton();
104:         endDateLabel = new javax.swing.JLabel();
105:         populationFileTextField = new javax.swing.JTextField();
106:         diabeticPopulationFileTextField = new javax.swing.JTextField();
107:         populationFileLabel = new javax.swing.JLabel();
108:         diabeticPopulationFileLabel = new javax.swing.JLabel();
109:         populationFileButton = new javax.swing.JButton();
110:         diabeticPopulationFileButton = new javax.swing.JButton();
111:         refAnaDateComboBox = new javax.swing.JComboBox();
112:         jLabel1 = new javax.swing.JLabel();
113:         centreIDComboBox = new javax.swing.JComboBox();
114:         startYearSpinner = new javax.swing.JSpinner();
115:         durationSpinner = new javax.swing.JSpinner();
116:
117:         setBorder(javax.swing.BorderFactory.createTitledBorder("Statistical Engine Configuration"));
118:
119:         centreIDLabel.setText("Centre ID *");
120:
121:         yearNowLabel.setText("Current year *");
122:         yearNowLabel.setToolTipText("last possible year for episode dates");
123:
124:         startDateLabel.setText("Start year *");
125:         startDateLabel.setToolTipText("first possible year when data are consistent and reliable");
126:
127:         refDateLabel.setText("Reference date *");
128:         refDateLabel.setToolTipText("date within chosen refyear to calculate age in the statistics");
129:
130:         yearNowSpinner.setModel(new SpinnerNumberModel(currentYear, currentYear - 100, currentYear + 100, 1));
131:         yearNowSpinner.setEditor(new JSpinner.NumberEditor(yearNowSpinner, "#"));
132:
133:         buttonPanel.setLayout(new java.awt.GridLayout(1, 0));
134:
```

```
135:     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
136:     jPanel1.setLayout(jPanel1Layout);
137:     jPanel1Layout.setHorizontalGroup(
138:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
139:             .addGap(0, 269, Short.MAX_VALUE)
140:     );
141:     jPanel1Layout.setVerticalGroup(
142:         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
143:             .addGap(0, 23, Short.MAX_VALUE)
144:     );
145:
146:     buttonPanel.add(jPanel1);
147:
148:     runStatisticalEngineButton.setText("Run Statistical Engine");
149:     runStatisticalEngineButton.addActionListener(new java.awt.event.ActionListener() {
150:         public void actionPerformed(java.awt.event.ActionEvent evt) {
151:             runStatisticalEngineButtonActionPerformed(evt);
152:         }
153:     });
154:
155:     javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
156:     jPanel2.setLayout(jPanel2Layout);
157:     jPanel2Layout.setHorizontalGroup(
158:         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
159:             .addComponent(runStatisticalEngineButton, javax.swing.GroupLayout.DEFAULT_SIZE, 269, Short.MAX_VALUE)
160:     );
161:     jPanel2Layout.setVerticalGroup(
162:         jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
163:             .addComponent(runStatisticalEngineButton)
164:     );
165:
166:     buttonPanel.add(jPanel2);
167:
168:     endDateLabel.setText("Duration (years) *");
169:
170:     populationFileLabel.setText("Population file *");
171:
172:     diabeticPopulationFileLabel.setText("Diabetic population file");
173:
174:     populationFileButton.setText("Browse");
175:     populationFileButton.addActionListener(new java.awt.event.ActionListener() {
176:         public void actionPerformed(java.awt.event.ActionEvent evt) {
177:             populationFileButtonActionPerformed(evt);
178:         }
179:     });
```

```
180:
181:     diabeticPopulationFileButton.setText("Browse");
182:     diabeticPopulationFileButton.addActionListener(new java.awt.event.ActionListener() {
183:         public void actionPerformed(java.awt.event.ActionEvent evt) {
184:             diabeticPopulationFileButtonActionPerformed(evt);
185:         }
186:     });
187:
188:     refAnaDateComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "30/06", "31/12" }));
189:
190:     jLabel1.setText("*= required fields");
191:
192:     centreIDComboBox.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "1=DARTS Dataset, Tayside
Scotland", "2=Umbria Dataset, Italy", "3=Healthgate Dataset, Austria", "4=Paulescu Datasets, Romania", "5=Norway Diabetes
Register", "6=Cyprus Diabetes Register", "7=Malta Diabetes Register" }));
193:
194:     startYearSpinner.setModel(new SpinnerNumberModel(currentYear, currentYear - 100, currentYear + 100,1));
195:     startYearSpinner.setEditor(new JSpinner.NumberEditor(startYearSpinner, "#"));
196:
197:     durationSpinner.setModel(new SpinnerNumberModel(duration, 1, 20, 1));
198:
199:     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
200:     this.setLayout(layout);
201:     layout.setHorizontalGroup(
202:         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
203:             .addComponent(buttonPanel, javax.swing.GroupLayout.DEFAULT_SIZE, 538, Short.MAX_VALUE)
204:             .addGroup(layout.createSequentialGroup()
205:                 .addComponent(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
206:                     .addGroup(layout.createSequentialGroup()
207:                         .addComponent(jLabel1, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 158, Short.MAX_VALUE)
208:                         .addGroup(javax.swing.GroupLayout.Alignment.LEADING,
layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
209:                             .addComponent(startDateLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
210:                             .addComponent(yearNowLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
211:                             .addComponent(centreIDLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 156, Short.MAX_VALUE)
212:                             .addComponent(populationFileLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
213:                             .addComponent(diabeticPopulationFileLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
214:                             .addComponent(refDateLabel, javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.PREFERRED_SIZE, 158, javax.swing.GroupLayout.PREFERRED_SIZE))
215:                     .addGap(0, 0, 0))
216:             .addContainerGap(10, Short.MAX_VALUE))
217:     );
```

```
215:                .addComponent(endDateLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 158,
javax.swing.GroupLayout.PREFERRED_SIZE))
216:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
217:                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
218:                .addComponent(refAnaDateComboBox, javax.swing.GroupLayout.Alignment.TRAILING, 0, 356,
Short.MAX_VALUE)
219:                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
220:                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
221:                .addComponent(diabeticPopulationFileTextField, javax.swing.GroupLayout.DEFAULT_SIZE,
283, Short.MAX_VALUE)
222:                .addComponent(populationFileTextField, javax.swing.GroupLayout.DEFAULT_SIZE, 283,
Short.MAX_VALUE))
223:                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
224:                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING, false)
225:                .addComponent(diabeticPopulationFileButton, 0, 0, Short.MAX_VALUE)
226:                .addComponent(populationFileButton))
227:                .addComponent(yearNowSpinner, javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.DEFAULT_SIZE, 356, Short.MAX_VALUE)
228:                .addComponent(centreIDComboBox, 0, 356, Short.MAX_VALUE)
229:                .addComponent(startYearSpinner, javax.swing.GroupLayout.DEFAULT_SIZE, 356, Short.MAX_VALUE)
230:                .addComponent(durationSpinner, javax.swing.GroupLayout.DEFAULT_SIZE, 356, Short.MAX_VALUE))
231:                .addContainerGap()
232:            );
233:
234:            layout.linkSize(javax.swing.SwingConstants.HORIZONTAL, new java.awt.Component[] {centreIDLabel,
diabeticPopulationFileLabel, endDateLabel, populationFileLabel, refDateLabel, startDateLabel, yearNowLabel});
235:
236:            layout.setVerticalGroup(
237:            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
238:            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING, layout.createSequentialGroup())
239:            .addContainerGap()
240:            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
241:            .addComponent(centreIDLabel, javax.swing.GroupLayout.PREFERRED_SIZE, 17,
javax.swing.GroupLayout.PREFERRED_SIZE)
242:            .addComponent(centreIDComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
243:            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
244:            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
245:            .addComponent(yearNowSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
246:            .addComponent(yearNowLabel))
247:            .addGap(8, 8, 8)
248:            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
249:            .addComponent(startDateLabel)
250:            .addComponent(startYearSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
```

```
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
251:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
252:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
253:             .addComponent(endDateLabel)
254:             .addComponent(durationSpinner, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
255:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
256:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
257:             .addComponent(refAnaDateComboBox, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
258:             .addComponent(refDateLabel))
259:         .addGap(10, 10, 10)
260:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
261:             .addComponent(populationFileButton)
262:             .addComponent(populationFileTextField, javax.swing.GroupLayout.PREFERRED_SIZE, 22,
javax.swing.GroupLayout.PREFERRED_SIZE)
263:             .addComponent(populationFileLabel))
264:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
265:         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
266:             .addComponent(diabeticPopulationFileButton)
267:             .addComponent(diabeticPopulationFileTextField, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
268:             .addComponent(diabeticPopulationFileLabel))
269:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
270:         .addComponent(jLabel1)
271:         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 123, Short.MAX_VALUE)
272:         .addComponent(buttonPanel, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
273:     );
274:
275:     layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[]
{diabeticPopulationFileButton, diabeticPopulationFileTextField});
276:
277:     layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[] {populationFileButton,
populationFileTextField});
278:
279:     layout.linkSize(javax.swing.SwingConstants.VERTICAL, new java.awt.Component[] {centreIDComboBox,
durationSpinner, refAnaDateComboBox, startYearSpinner, yearNowSpinner});
280:
281:     } // </editor-fold> // GEN-END: initComponents
282:     private void runStatisticalEngineButtonActionPerformed(java.awt.event.ActionEvent evt)
{ // GEN-FIRST: event_runStatisticalEngineButtonActionPerformed
283:         try {
284:
285:             checkConnection();
```

```
286:         if (populationFileTextField.getText().equals("")) {
287:             JOptionPane.showMessageDialog(this, "Fields marked with * cannot be empty", "WARNING",
JOptionPane.WARNING_MESSAGE);
288:         } else {
289:
290:
291:             //ROutputFrame.CreateAndShowFrame();
292:             SwingUtilities.invokeLater(new Runnable() {
293:
294:                 public void run() {
295:                     ROutputFrame r = new ROutputFrame(true);
296:                 }
297:             });
298:
299:             new Thread(new Runnable() {
300:
301:                 public void run() {
302:
303:                     try {
304:                         saveData();
305:                         statisticalEngineConfiguration.saveToFile(new
File(Configuration.statisticalEngineConfigurationPath));
306:                         SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy");
307:                         String command;
308:
309:                         String source = "\"" + Configuration.root + "_se_/source/r/main/biro_se.r" + "\"";
310:                         String dirse = "\"" + Configuration.root + "_se_" + "\"";
311:                         String dbformat = "\"postgres\"";
312:
313:                         String driverClass = "\"" +
bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getClassName() + "\"";
314:                         String jar =
(JarFinder.getCodeSource(Class.forName(bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getClassName()))).substring(5)
.replaceAll("%20", " ");
315:                         String classPath;
316:                         if (jar != null) {
317:                             classPath = "\"" +
(JarFinder.getCodeSource(Class.forName(bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getClassName()))).substring(5)
.replaceAll("%20", " ") + "\"";
318:
319:                         } else {
320:                             throw new Exception("I cannot find the Jar file containing the following class " +
bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getClassName());
321:                         }
322:                         String pathdb = "\"" + bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getUrl() +
```

```
"\n";
323:         String user = "\"" + bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getUser() +
"\n";
324:         String password = "\"" + bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getPwd()
+ "\n";
325:         String dbname = "\"" +
bIRODatabaseManagerConfiguration.getBIRODBMSDriver().getDatabaseName() + "\"";
326:         String dirdatastore = "\"" + "\"";
327:         String centreID = "\"" + statisticalEngineConfiguration.getCentreID() + "\"";
328:         String startdate = "";
329:         String enddate = "";
330:         int endYear = 0;
331:         if (statisticalEngineConfiguration.getRefAnaDate().equals("30/06")) {
332:             startdate = "\"" + "01/07/" + statisticalEngineConfiguration.getStartYear() + "\"";
333:             endYear = statisticalEngineConfiguration.getStartYear() +
statisticalEngineConfiguration.getDuration();
334:             enddate = "\"" + "30/06/" + endYear + "\"";
335:         } else if (statisticalEngineConfiguration.getRefAnaDate().equals("31/12")) {
336:             startdate = "\"" + "01/01/" + statisticalEngineConfiguration.getStartYear() + "\"";
337:             endYear = statisticalEngineConfiguration.getStartYear() +
statisticalEngineConfiguration.getDuration() - 1;
338:             enddate = "\"" + "31/12/" + endYear + "\"";
339:         }
340:         int yearnow = statisticalEngineConfiguration.getYearNow();
341:         String refanadate = "\"" + statisticalEngineConfiguration.getRefAnaDate() + "\"";
342:         String filepop = "\"" +
statisticalEngineConfiguration.getPopulationFilePath().replaceAll("\\\\", "/") + "\"";
343:         String filepopdiab = "\"" +
statisticalEngineConfiguration.getDiabeticPopulationFilePath() + "\"";
344:         String activitytable = "";
345:         if (activitytablePresent) {
346:             activitytable = "1";
347:         } else {
348:             activitytable = "0";
349:         }
350:         int cex = 2;
351:
352:
353:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
354:         MyEngine re = MyEngine.getInstance();
355:
356:
357:
358:         System.out.println("Statistical engine is loading data. Please wait. " + "\n");
359:         command = "rm(list=ls())";
```

```
360:         System.out.println("\nI'm executing the following command:" + "\n");
361:         System.out.println(command + "\n");
362:         re.eval(command);
363:
364:         command = "source(" + source + ")";
365:         System.out.println("\nI'm executing the following command:" + "\n");
366:         System.out.println(command + "\n");
367:         re.eval(command);
368:
369:         command = "sinkfile <- file(\"" + Configuration.root +
"_se_/statisticalEngineSinkFile.txt" + "\"" + ", open=\"wt\")";
370:         //System.out.println("\nI'm executing the following command:" + "\n");
371:         //System.out.println(command + "\n");
372:         re.eval(command);
373:
374:         command = "sink(sinkfile)";
375:         //System.out.println("\nI'm executing the following command:" + "\n");
376:         //System.out.println(command + "\n");
377:         re.eval(command);
378:
379:         command = "sink(sinkfile, type=\"message\")";
380:         System.out.println("\nI'm executing the following command:" + "\n");
381:         System.out.println(command + "\n");
382:         re.eval(command);
383:
384:         command = "BIRO_se(dirse=" + dirse +
385:         ", dbformat=" + dbformat +
386:         ", driverClass=" + driverClass +
387:         ", classPath=" + classPath +
388:         ", identifier.quote=\"\\\"\" +
389:         ", pathdb=" + pathdb +
390:         ", user=" + user +
391:         ", password=" + password +
392:         ", dbname=" + dbname +
393:         ", dirdatastore=" + dirdatastore +
394:         ", centre_id=" + centreID +
395:         ", startdate=" + startdate +
396:         ", enddate=" + enddate +
397:         ", yearnow=" + yearnow +
398:         ", refanadate=" + refanadate +
399:         ", logfile = \"statisticalEngine.log\" +
400:         ", cex= 1" +
401:         ", wide=1" +
402:         ", filepop=" + filepop +
403:         ", filepopdiab=" + filepopdiab +
```

```
404:         ", activitytable=" + activitytable + "));
405:
406:         System.out.println("\nI'm executing the following command:" + "\n\n");
407:         System.out.println(command + "\n");
408:         re.eval(command);
409:
410:         command = "unlink(\"" + Configuration.root + "_se_/statisticalEngineSinkFile.txt" +
"\n\"" + " + " + "));";
411:         //System.out.println("\nI'm executing the following command:" + "\n\n");
412:         //System.out.println(command + "\n");
413:         re.eval(command);
414:
415:         FileWriter aWriter = new FileWriter(Configuration.root +
_se_/statisticalEngineSinkFile.txt", true);
416:         aWriter.write("\nStatistical engine process is finished");
417:         aWriter.close();
418:
419:         //re.end();
420:
421:         /*
422:         for (String s : new File(Configuration.root + "_se_/output/data").list()) {
423:         if (!communicationSoftwareConfiguration.containsStatisticalObject(s)) {
424:         communicationSoftwareConfiguration.getStatisticalObjects().add(new
StatisticalObject(s,
statisticalEngineConfiguration.getCentreID(),statisticalEngineConfiguration.getStartYear(),statisticalEngineConfiguration.
getDuration()));
425:         }
426:         }
427:         */
428:         setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
429:
430:
431:         } catch (Exception ex) {
432:             System.out.println("An error occurred during statistical engine process:" + "\n\n");
433:             System.out.println(ex.getMessage());
434:             Logger.getLogger(StatisticalEnginePanel.class.getName()).log(Level.SEVERE, null, ex);
435:         }
436:     }
437:     }).start();
438: }
439: } catch (Exception e) {
440:     JOptionPane.showMessageDialog(this, e.getMessage(), "WARNING", JOptionPane.WARNING_MESSAGE);
441: }
442: } //GEN-LAST:event_runStatisticalEngineButtonActionPerformed
443:
```

```
444:     private void populationFileButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_populationFileButtonActionPerformed
445:
446:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
447:         SwingUtilities.invokeLater(new Runnable() {
448:
449:             public void run() {
450:                 JFileChooser fc = new JFileChooser();
451:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("File CSV", "csv"));
452:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
453:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
454:                 fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
455:                 fc.setSelectedFile(new File(statisticalEngineConfiguration.getPopulationFilePath()));
456:                 int returnVal = fc.showOpenDialog(StatisticalEnginePanel.this);
457:                 if (returnVal == JFileChooser.APPROVE_OPTION) {
458:                     File file = fc.getSelectedFile();
459:                     populationFileTextField.setText(file.getAbsolutePath());
460:                 }
461:                 setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
462:             }
463:         });
464:     } //GEN-LAST:event_populationFileButtonActionPerformed
465:
466:     private void diabeticPopulationFileButtonActionPerformed(java.awt.event.ActionEvent evt)
{ //GEN-FIRST:event_diabeticPopulationFileButtonActionPerformed
467:         setCursor(new Cursor(Cursor.WAIT_CURSOR));
468:         SwingUtilities.invokeLater(new Runnable() {
469:
470:             public void run() {
471:                 JFileChooser fc = new JFileChooser();
472:                 fc.addChoosableFileFilter(new FileNameExtensionFilter("File CSV", "csv"));
473:                 FileFilter acceptAllFileFilter = fc.getAcceptAllFileFilter();
474:                 fc.removeChoosableFileFilter(acceptAllFileFilter);
475:                 fc.setFileSelectionMode(JFileChooser.FILES_ONLY);
476:                 fc.setSelectedFile(new File(statisticalEngineConfiguration.getDiabeticPopulationFilePath()));
477:                 int returnVal = fc.showOpenDialog(StatisticalEnginePanel.this);
478:                 if (returnVal == JFileChooser.APPROVE_OPTION) {
479:                     File file = fc.getSelectedFile();
480:                     diabeticPopulationFileTextField.setText(file.getAbsolutePath());
481:                 }
482:                 setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
483:             }
484:         });
485:     } //GEN-LAST:event_diabeticPopulationFileButtonActionPerformed
486:
```

```
487: public void loadData() {
488:     centreIDComboBox.setSelectedIndex(statisticalEngineConfiguration.getCentreID() - 1);
489:     yearNowSpinner.setValue(statisticalEngineConfiguration.getYearNow());
490:     startYearSpinner.setValue(statisticalEngineConfiguration.getStartYear());
491:     durationSpinner.setValue(statisticalEngineConfiguration.getDuration());
492:     if (statisticalEngineConfiguration.getRefAnaDate() != null) {
493:         refAnaDateComboBox.setSelectedItem(statisticalEngineConfiguration.getRefAnaDate());
494:     } else {
495:         refAnaDateComboBox.setSelectedIndex(0);
496:     }
497:     populationFileTextField.setText(statisticalEngineConfiguration.getPopulationFilePath());
498:     diabeticPopulationFileTextField.setText(statisticalEngineConfiguration.getDiabeticPopulationFilePath());
499: }
500:
501:
502: @Override
503: public void saveData() throws Exception {
504:
505:     statisticalEngineConfiguration.setCentreID(centreIDComboBox.getSelectedIndex() + 1);
506:     statisticalEngineConfiguration.setYearNow(((Integer) (yearNowSpinner.getValue())).intValue());
507:     statisticalEngineConfiguration.setStartYear(((Integer) (startYearSpinner.getValue())).intValue());
508:     statisticalEngineConfiguration.setDuration(((Integer) (durationSpinner.getValue())).intValue());
509:     statisticalEngineConfiguration.setRefAnaDate((String) refAnaDateComboBox.getSelectedItem());
510:     statisticalEngineConfiguration.setDiabeticPopulationFilePath(diabeticPopulationFileTextField.getText());
511:     statisticalEngineConfiguration.setPopulationFilePath(populationFileTextField.getText());
512: }
513:
514: private void checkConnection() throws Exception {
515:     try {
516:         saveData();
517:         statisticalEngineConfiguration.saveToFile(new File(Configuration.statisticalEngineConfigurationPath));
518:         ConnectionManager manager =
ConnectionManager.initializeManager(bIRODatabaseManagerConfiguration.getBIRODBMSDriver());
519:         manager.connect();
520:         activitytablePresent = (manager.executeQuery("select * from activity_data")).next();
521:     } catch (Exception e) {
522:         throw new Exception("Something went wrong when connecting to BIRO database. Plesae check the
configuration");
523:     }
524: }
525: // Variables declaration - do not modify//GEN-BEGIN:variables
526: private javax.swing.JPanel buttonPanel;
527: private javax.swing.JComboBox centreIDComboBox;
528: private javax.swing.JLabel centreIDLabel;
529: private javax.swing.JButton diabeticPopulationFileButton;
```

```
530: private javax.swing.JLabel diabeticPopulationFileLabel;
531: private javax.swing.JTextField diabeticPopulationFileTextField;
532: private javax.swing.JSpinner durationSpinner;
533: private javax.swing.JLabel endDateLabel;
534: private javax.swing.JLabel jLabel1;
535: private javax.swing.JPanel jPanel1;
536: private javax.swing.JPanel jPanel2;
537: private javax.swing.JButton populationFileButton;
538: private javax.swing.JLabel populationFileLabel;
539: private javax.swing.JTextField populationFileTextField;
540: private javax.swing.JComboBox refAnaDateComboBox;
541: private javax.swing.JLabel refDateLabel;
542: private javax.swing.JButton runStatisticalEngineButton;
543: private javax.swing.ButtonGroup sourceButtonGroup;
544: private javax.swing.JLabel startDateLabel;
545: private javax.swing.JSpinner startYearSpinner;
546: private javax.swing.JLabel yearNowLabel;
547: private javax.swing.JSpinner yearNowSpinner;
548: // End of variables declaration//GEN-END:variables
549: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * The BIRO-Installer is part of WP Technology Transfer of the BIRO Project.
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.utils;
31:
32: import eu.biro.adaptor2.DBMSDriver;
33: import java.sql.Connection;
34: import java.sql.DatabaseMetaData;
35: import java.sql.PreparedStatement;
36: import java.sql.ResultSet;
37: import java.sql.SQLException;
38: import java.sql.Statement;
39:
40: import org.plibrary.PObject;
41:
42: public final class ConnectionManager
43:     extends PObject
44:     implements Runnable {
45:
```

```
46: private static final ConnectionManager manager = new ConnectionManager();
47:
48: private DBMSDriver driver;
49: private Connection conn;
50:
51: private ConnectionManager() {
52:     Runtime.getRuntime().addShutdownHook(new Thread(this));
53: }
54:
55: public void run() {
56:     try {
57:         close();
58:     } catch (Exception e) {
59:         e.printStackTrace();
60:     }
61: }
62:
63: public static ConnectionManager getManager() {
64:     return manager;
65: }
66:
67: /**
68:  * Init the driver, must be called before anything else
69:  */
70:
71:
72: public static ConnectionManager initializeManager(DBMSDriver driver) throws ClassNotFoundException,
SQLException, Exception {
73:     if (manager != null)
74:         manager.close();
75:     // manager = new ConnectionManager();
76:     manager.driver = driver;
77:     driver.initialize();
78:     return manager;
79: }
80:
81: public synchronized void close() throws SQLException,Exception {
82:     if (conn != null && !conn.isClosed()) {
83:         driver.close(getStatement());
84:         conn.close();
85:         conn = null;
86:     }
87: }
88:
89: public synchronized void connect() throws Exception {
```

```
90:         if (conn == null || conn.isClosed())
91:             conn = driver.openConnection();
92:     }
93:
94:     private synchronized Statement getStatement() throws Exception {
95:         connect();
96:         return conn.createStatement();
97:     }
98:
99:     private synchronized PreparedStatement getPreparedStatement(String query) throws SQLException, Exception {
100:         connect();
101:         return conn.prepareStatement(query);
102:     }
103:
104:     public ResultSet executeQuery(String query) throws SQLException, Exception {
105:         return getStatement().executeQuery(query);
106:     }
107:
108:     int executeUpdate(String query) throws SQLException, Exception {
109:         return getStatement().executeUpdate(query);
110:     }
111:
112:     PreparedStatement prepareStatement(String query) throws SQLException,Exception {
113:         return getPreparedStatement(query);
114:     }
115:
116:     public boolean doesExistTable(String name) throws SQLException, Exception {
117:         connect();
118:         DatabaseMetaData dbm = conn.getMetaData();
119:         ResultSet r = dbm.getTables(null, null, name, null);
120:         boolean answ = r.next();
121:         r.close();
122:         return answ;
123:     }
124:
125:     public DatabaseMetaData getMetaData() throws SQLException {
126:         return conn.getMetaData();
127:     }
128:
129:     public DBMSDriver getDriver() {
130:         return driver;
131:     }
132: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * The BIRO-Installer is part of WP Technology Transfer of the BIRO Project.
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.utils;
31:
32: import eu.biro.adaptor2.DBMSDriver;
33: import java.io.IOException;
34: import java.lang.reflect.Method;
35: import java.net.MalformedURLException;
36: import java.net.URL;
37: import java.net.URLClassLoader;
38: import java.sql.Connection;
39: import java.sql.DriverManager;
40: import java.sql.SQLException;
41:
42:
43: public class CustomDBMSDriver extends DBMSDriver {
44:
45:     private String urlPattern;
```

```
46: private String jarPath;
47:
48: public CustomDBMSDriver(String dbName, String className, String urlPattern, String jarPath) {
49:     super(dbName, className, urlPattern.substring(0, urlPattern.indexOf("<hostandport>")));
50:     this.urlPattern = urlPattern;
51:     this.jarPath = jarPath;
52: }
53: public Connection getConnection() throws SQLException, MalformedURLException {
54:     String url = "";
55:     Connection c = null;
56:     URL jarFileLoaderURL[] = {};
57:     // if (Pattern.matches(".*<hostandport>.*<database>.*<user>.*<password>.*", this.urlPattern)) {
58:     if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") && urlPattern.contains(
"<username>") && urlPattern.contains("<password>")) {
59:         url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
60:         url = url.replaceAll("<database>", getDatabaseName());
61:         url = url.replaceAll("<user>", getUser());
62:         url = url.replaceAll("<password>", getPwd());
63:         c = DriverManager.getConnection(url, getUser(), getPwd());
64:         //} else if (Pattern.matches("[.]<hostandport>[.]<database>[.]", this.urlPattern)) {
65:         } else if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") &&
!urlPattern.contains("<username>") && !urlPattern.contains("<password>")) {
66:             url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
67:             url = url.replaceAll("<database>", getDatabaseName());
68:             c = DriverManager.getConnection(url, getUser(), getPwd());
69:         } else {
70:             c = DriverManager.getConnection(urlPattern);
71:         }
72:         return c;
73:     }
74:
75: @Override
76: public Connection openConnection() throws SQLException, MalformedURLException {
77:     String url = "";
78:     Connection c = null;
79:
80:     //if (Pattern.matches(".*<hostandport>.*<database>.*<user>.*<password>.*", this.urlPattern)) {
81:     if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") && urlPattern.contains(
"<username>") && urlPattern.contains("<password>")) {
82:         url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
83:         url = url.replaceAll("<database>", getDatabaseName());
84:         url = url.replaceAll("<user>", getUser());
85:         url = url.replaceAll("<password>", getPwd());
86:         c = DriverManager.getConnection(url, getUser(), getPwd());
87:         // } else if (Pattern.matches("[.]<hostandport>[.]<database>[.]", this.urlPattern)) {
```

```
88:         } else if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") &&
!urlPattern.contains("<username>") && !urlPattern.contains("<password>")) {
89:             url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
90:             url = url.replaceAll("<database>", getDatabaseName());
91:             c = DriverManager.getConnection(url, getUser(), getPwd());
92:         } else {
93:             c = DriverManager.getConnection(urlPattern);
94:         }
95:         return c;
96:     }
97:
98:     @Override
99:     public String getUrl() {
100:         String url = "";
101:         //if (Pattern.matches(".*<hostandport>.*<database>.*<user>.*<password>.*", this.urlPattern)) {
102:         if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") && urlPattern.contains(
"<username>") && urlPattern.contains("<password>")) {
103:             url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
104:             url = url.replaceAll("<database>", getDatabaseName());
105:             url = url.replaceAll("<user>", getUser());
106:             url = url.replaceAll("<password>", getPwd());
107:
108:             // } else if (Pattern.matches("[.]<hostandport>[.]<database>[.]", this.urlPattern)) {
109:             } else if (urlPattern.contains("<hostandport>") && urlPattern.contains("<database>") &&
!urlPattern.contains("<username>") && !urlPattern.contains("<password>")) {
110:                 url = urlPattern.replaceAll("<hostandport>", getHostAndPort());
111:                 url = url.replaceAll("<database>", getDatabaseName());
112:             } else {
113:                 url = urlPattern;
114:             }
115:             return url;
116:         }
117:
118:         @Override
119:         public void initialize() throws Exception {
120:             URLClassLoader sysLoader = (URLClassLoader) ClassLoader.getSystemClassLoader();
121:             URL urls[] = sysLoader.getURLs();
122:             URL u = new URL("file:/// " + this.jarPath);
123:             for (int i = 0; i < urls.length; i++) {
124:                 if ((urls[i].toString()).equalsIgnoreCase(u.toString())) {
125:                     //System.out.println("URL" + u + "is already in the CLASSPATH"); //useful for DEBUG
126:                     return;
127:                 }
128:             }
129:             Class sysclass = URLClassLoader.class;
```

```
130:         try {
131:             Method method = sysclass.getDeclaredMethod("addURL", URL.class);
132:             method.setAccessible(true);
133:             method.invoke(sysLoader, new Object[]{u});
134:         } catch (Throwable t) {
135:             t.printStackTrace();
136:             throw new IOException("Error, could not add URL to system classloader");
137:         }
138:         urls = sysLoader.getURLs();
139:         Class.forName(getClassName());
140:     }
141:
142: }
143:
144:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      AbstractValueMapping.java
5:  * Authors:
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  *
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31:
32: package eu.biro.birobox.utils;
33:
34: import java.io.File;
35: import java.io.IOException;
36:
37: public interface FileListener
38: {
39:     /**
40:      * Called when one of the monitored files are created, deleted
41:      * or modified.
42:      *
43:      * @param file  File which has been changed.
44:      */
45:     void fileChanged (File file)throws IOException;
```

07/15/09
13:17:36

BIROBox/src/eu/biro/birobox/utils/FileListener.java

2

```
46: }  
47:  
48:  
49:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      AbstractValueMapping.java
5:  * Authors:
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10: * the Free Software Foundation; either version 2, or (at your option)
11: * any later version.
12: *
13: * This file is distributed in the hope that it will be useful,
14: * but WITHOUT ANY WARRANTY; without even the implied warranty of
15: * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16: * GNU General Public License for more details.
17: *
18: * You should have received a copy of the GNU General Public License
19: * along with this file; see the file COPYING. If not, write to
20: * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21: *
22: * In short: you may use this file any way you like, as long as you
23: * don't charge money for it, remove this notice, or hold anyone liable
24: * for its results.
25: *
26:
27: * GPL Copyright, The BIRO Project
28: *
29: **/
30: package eu.biro.birobox.utils;
31:
32:
33: import eu.biro.birobox.utils.FileListener;
34: import java.io.IOException;
35: import java.util.*;
36: import java.io.File;
37: import java.lang.ref.WeakReference;
38: import java.util.logging.Level;
39: import java.util.logging.Logger;
40:
41: /**
42:  * Class for monitoring changes in disk files.
43:  * Usage:
44:  *
45:  * 1. Implement the FileListener interface.
```

```
46: *      2. Create a FileMonitor instance.
47: *      3. Add the file(s)/directory(ies) to listen for.
48: *
49: * fileChanged() will be called when a monitored file is created,
50: * deleted or its modified time changes.
51: *
52: * @author <a href="mailto:jacob.dreyer@geosoft.no">Jacob Dreyer</a>
53: */
54: public class FileMonitor {
55:
56:     private Timer timer_;
57:     private HashMap files_;           // File -> Long
58:     private Collection listeners_;    // of WeakReference(FileListener)
59:
60:     /**
61:      * Create a file monitor instance with specified polling interval.
62:      *
63:      * @param pollingInterval Polling interval in milli seconds.
64:      */
65:     public FileMonitor(long pollingInterval) {
66:         files_ = new HashMap();
67:         listeners_ = new ArrayList();
68:
69:         timer_ = new Timer(true);
70:         timer_.schedule(new FileMonitorNotifier(), 0, pollingInterval);
71:     }
72:
73:     /**
74:      * Stop the file monitor polling.
75:      */
76:     public void stop() {
77:         timer_.cancel();
78:     }
79:
80:     /**
81:      * Add file to listen for. File may be any java.io.File (including a
82:      * directory) and may well be a non-existing file in the case where the
83:      * creating of the file is to be trepped.
84:      * <p>
85:      * More than one file can be listened for. When the specified file is
86:      * created, modified or deleted, listeners are notified.
87:      *
88:      * @param file File to listen for.
89:      */
90:     public void addFile(File file) {
```

```
91:         if (!files_.containsKey(file)) {
92:             long modifiedTime = file.exists() ? file.lastModified() : -1;
93:             files_.put(file, new Long(modifiedTime));
94:         }
95:     }
96:
97:     /**
98:      * Remove specified file for listening.
99:      *
100:     * @param file File to remove.
101:     */
102:     public void removeFile(File file) {
103:         files_.remove(file);
104:     }
105:
106:     /**
107:      * Add listener to this file monitor.
108:      *
109:     * @param fileListener Listener to add.
110:     */
111:     public void addListener(FileListener fileListener) {
112:         // Don't add if its already there
113:         for (Iterator i = listeners_.iterator(); i.hasNext();) {
114:             WeakReference reference = (WeakReference) i.next();
115:             FileListener listener = (FileListener) reference.get();
116:             if (listener == fileListener) {
117:                 return;
118:             }
119:         }
120:
121:         // Use WeakReference to avoid memory leak if this becomes the
122:         // sole reference to the object.
123:         listeners_.add(new WeakReference(fileListener));
124:     }
125:
126:     /**
127:      * Remove listener from this file monitor.
128:      *
129:     * @param fileListener Listener to remove.
130:     */
131:     public void removeListener(FileListener fileListener) {
132:         for (Iterator i = listeners_.iterator(); i.hasNext();) {
133:             WeakReference reference = (WeakReference) i.next();
134:             FileListener listener = (FileListener) reference.get();
135:             if (listener == fileListener) {
```

```
136:         i.remove();
137:         break;
138:     }
139: }
140: }
141:
142: /**
143:  * This is the timer thread which is executed every n milliseconds
144:  * according to the setting of the file monitor. It investigates the
145:  * file in question and notify listeners if changed.
146:  */
147: private class FileMonitorNotifier extends TimerTask {
148:
149:     public void run(){
150:         // Loop over the registered files and see which have changed.
151:         // Use a copy of the list in case listener wants to alter the
152:         // list within its fileChanged method.
153:         Collection files = new ArrayList(files_.keySet());
154:
155:         for (Iterator i = files.iterator(); i.hasNext();) {
156:             File file = (File) i.next();
157:             long lastModifiedTime = ((Long) files_.get(file)).longValue();
158:             long newModifiedTime = file.exists() ? file.lastModified() : -1;
159:
160:             // Chek if file has changed
161:             if (newModifiedTime != lastModifiedTime) {
162:
163:                 // Register new modified time
164:                 files_.put(file, new Long(newModifiedTime));
165:
166:                 // Notify listeners
167:                 for (Iterator j = listeners_.iterator(); j.hasNext();) {
168:                     WeakReference reference = (WeakReference) j.next();
169:                     FileListener listener = (FileListener) reference.get();
170:
171:                     // Remove from list if the back-end object has been GC'd
172:                     if (listener == null) {
173:                         j.remove();
174:                     } else {
175:                         try {
176:                             listener.fileChanged(file);
177:                         } catch (IOException ex) {
178:                             Logger.getLogger(FileMonitor.class.getName()).log(Level.SEVERE, null, ex);
179:                         }
180:                     }
181:                 }
182:             }
183:         }
184:     }
185: }
```

```
181:         }
182:     }
183: }
184: }
185: }
186:
187: private class TestListener
188:     implements FileListener {
189:
190:     public void fileChanged(File file) {
191:         System.out.println("File changed: " + file);
192:     }
193: }
194: }
195:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:   Pietro Palladino, Valentina Baglioni
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * The BIRO-Installer is part of WP Technology Transfer of the BIRO Project.
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30: package eu.biro.birobox.utils;
31:
32: import java.io.File;
33: import java.net.URL;
34: import java.io.IOException;
35: import java.net.URLClassLoader;
36: import java.net.MalformedURLException;
37:
38: public class JarFileLoader extends URLClassLoader
39: {
40:     public JarFileLoader (URL[] urls)
41:     {
42:         super (urls);
43:     }
44:
45:     public void addFile (String path) throws MalformedURLException
```

```
46:     {
47:         String urlPath = "file://" + path;
48:         addURL (new URL (urlPath));
49:     }
50:
51:     public static void main (String args [])
52:     {
53:         try
54:         {
55:             System.out.println ("First attempt...");
56:             Class.forName ("org.postgresql.Driver");
57:         }
58:         catch (Exception ex)
59:         {
60:             System.out.println ("Failed.");
61:         }
62:
63:         try
64:         {
65:             URL urls [] = {};
66:
67:             JarFileLoader cl = new JarFileLoader (urls);
68:             File f =new File("");
69:             System.out.println(f.getAbsolutePath());
70:             System.out.println(f.getCanonicalPath());
71:             cl.addFile ("C:/Documents and
Settings/Valentina/Documenti/NetBeansProjects/BIROBox/libraries/postgresql-8.2-504.jdbc3.jar");
72:             System.out.println ("Second attempt...");
73:             cl.loadClass ("org.postgresql.Driver");
74:             System.out.println ("Success!");
75:         }
76:         catch (Exception ex)
77:         {
78:             System.out.println ("Failed.");
79:             ex.printStackTrace ();
80:         }
81:     }
82: }
83:
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * The BIRO-Installer is part of WP Technology Transfer of the BIRO Project.
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.utils;
32:
33: import java.io.IOException;
34: import java.net.JarURLConnection;
35: import java.net.URL;
36: import java.net.URLConnection;
37:
38: public class JarFinder {
39:
40:     /** Method returns code source of given class.
41:      * This is URL of classpath folder, zip or jar file.
42:      * If code source is unknown, returns null (for example, for classes java.io.*).
43:      * @param clazz For example, java.sql.SQLException.class¹
44:      * @return for example, "file:/C:/jdev10/jdev/mywork/classes/"
45:      * or "file:/C:/works/projects/classes12.zip"
```

```
46:      */
47:      public static String getCodeSource(Class clazz) {
48:          if (clazz == null || clazz.getProtectionDomain() == null || clazz.getProtectionDomain().getCodeSource() ==
null || clazz.getProtectionDomain().getCodeSource().getLocation() == null) // This typically happens for system
classloader // (java.lang.* etc. classes)
49:          {
50:              return null;
51:          }
52:          System.out.println(clazz.getProtectionDomain().getCodeSource().getLocation().toString());
53:          return clazz.getProtectionDomain().getCodeSource().getLocation().toString();
54:      }
55:  }
56:
57:  public static URL getJarURL(Class c) {
58:      URL clsUrl = c.getResource(c.getSimpleName() + ".class");
59:      //URL clsUrl = getClass().getResource(getClass().getSimpleName() + ".class");
60:      if (clsUrl != null) {
61:          try {
62:              URLConnection conn = clsUrl.openConnection();
63:              if (conn instanceof JarURLConnection) {
64:                  JarURLConnection connection = (JarURLConnection) conn;
65:                  return connection.getJarFileURL();
66:              }
67:          } catch (IOException e) {
68:              throw new RuntimeException(e);
69:          }
70:      }
71:      return null;
72:  }
73: }
```

```
1: /**
2:  * Project: BIRO-Project (Funded by European Commission 2005-2008)
3:  *
4:  * File:      StatisticalEnginePanel.java
5:  * Authors:
6:  *
7:  * COPYRIGHT INFORMATION
8:  * This file is free software; you can redistribute it and/or modify
9:  * it under the terms of the GNU General Public License as published by
10:  * the Free Software Foundation; either version 2, or (at your option)
11:  * any later version.
12:  *
13:  * This file is distributed in the hope that it will be useful,
14:  * but WITHOUT ANY WARRANTY; without even the implied warranty of
15:  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
16:  * GNU General Public License for more details.
17:  *
18:  * You should have received a copy of the GNU General Public License
19:  * along with this file; see the file COPYING. If not, write to
20:  * the Free Software Foundation, 675 Mass Ave, Cambridge, MA 02139, USA.
21:  *
22:  * In short: you may use this file any way you like, as long as you
23:  * don't charge money for it, remove this notice, or hold anyone liable
24:  * for its results.
25:  *
26:  * The BIRO-Installer is part of WP Technology Transfer of the BIRO Project.
27:  * GPL Copyright, The BIRO Project
28:  *
29:  */
30:
31: package eu.biro.birobox.utis;
32:
33: import javax.swing.*;
34: import javax.swing.SpringLayout;
35: import java.awt.*;
36:
37: /**
38:  * A 1.4 file that provides utility methods for
39:  * creating form- or grid-style layouts with SpringLayout.
40:  * These utilities are used by several programs, such as
41:  * SpringBox and SpringCompactGrid.
42:  */
43: public class SpringUtilities {
44:     /**
45:      * A debugging utility that prints to stdout the component's
```

```
46:      * minimum, preferred, and maximum sizes.
47:      */
48:  public static void printSizes(Component c) {
49:      System.out.println("minimumSize = " + c.getMinimumSize());
50:      System.out.println("preferredSize = " + c.getPreferredSize());
51:      System.out.println("maximumSize = " + c.getMaximumSize());
52:  }
53:
54:  /**
55:   * Aligns the first <code>rows</code> * <code>cols</code>
56:   * components of <code>parent</code> in
57:   * a grid. Each component is as big as the maximum
58:   * preferred width and height of the components.
59:   * The parent is made just big enough to fit them all.
60:   *
61:   * @param rows number of rows
62:   * @param cols number of columns
63:   * @param initialX x location to start the grid at
64:   * @param initialY y location to start the grid at
65:   * @param xPad x padding between cells
66:   * @param yPad y padding between cells
67:   */
68:  public static void makeGrid(Container parent,
69:                              int rows, int cols,
70:                              int initialX, int initialY,
71:                              int xPad, int yPad) {
72:      SpringLayout layout;
73:      try {
74:          layout = (SpringLayout)parent.getLayout();
75:      } catch (ClassCastException exc) {
76:          System.err.println("The first argument to makeGrid must use SpringLayout.");
77:          return;
78:      }
79:
80:      Spring xPadSpring = Spring.constant(xPad);
81:      Spring yPadSpring = Spring.constant(yPad);
82:      Spring initialXSpring = Spring.constant(initialX);
83:      Spring initialYSpring = Spring.constant(initialY);
84:      int max = rows * cols;
85:
86:      //Calculate Springs that are the max of the width/height so that all
87:      //cells have the same size.
88:      Spring maxWidthSpring = layout.getConstraints(parent.getComponent(0)).
89:                              getWidth();
90:      Spring maxHeightSpring = layout.getConstraints(parent.getComponent(0)).
```

```
91:             getWidth();
92:     for (int i = 1; i < max; i++) {
93:         SpringLayout.Constraints cons = layout.getConstraints(
94:             parent.getComponent(i));
95:
96:         maxWidthSpring = Spring.max(maxWidthSpring, cons.getWidth());
97:         maxHeightSpring = Spring.max(maxHeightSpring, cons.getHeight());
98:     }
99:
100:    //Apply the new width/height Spring. This forces all the
101:    //components to have the same size.
102:    for (int i = 0; i < max; i++) {
103:        SpringLayout.Constraints cons = layout.getConstraints(
104:            parent.getComponent(i));
105:
106:        cons.setWidth(maxWidthSpring);
107:        cons.setHeight(maxHeightSpring);
108:    }
109:
110:    //Then adjust the x/y constraints of all the cells so that they
111:    //are aligned in a grid.
112:    SpringLayout.Constraints lastCons = null;
113:    SpringLayout.Constraints lastRowCons = null;
114:    for (int i = 0; i < max; i++) {
115:        SpringLayout.Constraints cons = layout.getConstraints(
116:            parent.getComponent(i));
117:
118:        if (i % cols == 0) { //start of new row
119:            lastRowCons = lastCons;
120:            cons.setX(initialXSpring);
121:        } else { //x position depends on previous component
122:            cons.setX(Spring.sum(lastCons.getConstraint(SpringLayout.EAST),
123:                xPadSpring));
124:        }
125:
126:        if (i / cols == 0) { //first row
127:            cons.setY(initialYSpring);
128:        } else { //y position depends on previous row
129:            cons.setY(Spring.sum(lastRowCons.getConstraint(SpringLayout.SOUTH),
130:                yPadSpring));
131:        }
132:        lastCons = cons;
133:    }
134:
135:    //Set the parent's size.
    SpringLayout.Constraints pCons = layout.getConstraints(parent);
```

```
136:         pCons.setConstraint(SpringLayout.SOUTH,
137:                               Spring.sum(
138:                                   Spring.constant(yPad),
139:                                   lastCons.getConstraint(SpringLayout.SOUTH)));
140:         pCons.setConstraint(SpringLayout.EAST,
141:                               Spring.sum(
142:                                   Spring.constant(xPad),
143:                                   lastCons.getConstraint(SpringLayout.EAST)));
144:     }
145:
146:     /* Used by makeCompactGrid. */
147:     private static SpringLayout.Constraints getConstraintsForCell(
148:         int row, int col,
149:         Container parent,
150:         int cols) {
151:         SpringLayout layout = (SpringLayout) parent.getLayout();
152:         Component c = parent.getComponent(row * cols + col);
153:         return layout.getConstraints(c);
154:     }
155:
156:     /**
157:     * Aligns the first <code>rows</code> * <code>cols</code>
158:     * components of <code>parent</code> in
159:     * a grid. Each component in a column is as wide as the maximum
160:     * preferred width of the components in that column;
161:     * height is similarly determined for each row.
162:     * The parent is made just big enough to fit them all.
163:     *
164:     * @param rows number of rows
165:     * @param cols number of columns
166:     * @param initialX x location to start the grid at
167:     * @param initialY y location to start the grid at
168:     * @param xPad x padding between cells
169:     * @param yPad y padding between cells
170:     */
171:     public static void makeCompactGrid(Container parent,
172:                                       int rows, int cols,
173:                                       int initialX, int initialY,
174:                                       int xPad, int yPad) {
175:         SpringLayout layout;
176:         try {
177:             layout = (SpringLayout)parent.getLayout();
178:         } catch (ClassCastException exc) {
179:             System.err.println("The first argument to makeCompactGrid must use SpringLayout.");
180:             return;
```

```
181:     }
182:
183:     //Align all cells in each column and make them the same width.
184:     Spring x = Spring.constant(initialX);
185:     for (int c = 0; c < cols; c++) {
186:         Spring width = Spring.constant(0);
187:         for (int r = 0; r < rows; r++) {
188:             width = Spring.max(width,
189:                 getConstraintsForCell(r, c, parent, cols).
190:                 getWidth());
191:         }
192:         for (int r = 0; r < rows; r++) {
193:             SpringLayout.Constraints constraints =
194:                 getConstraintsForCell(r, c, parent, cols);
195:             constraints.setX(x);
196:             constraints.setWidth(width);
197:         }
198:         x = Spring.sum(x, Spring.sum(width, Spring.constant(xPad)));
199:     }
200:
201:     //Align all cells in each row and make them the same height.
202:     Spring y = Spring.constant(initialY);
203:     for (int r = 0; r < rows; r++) {
204:         Spring height = Spring.constant(0);
205:         for (int c = 0; c < cols; c++) {
206:             height = Spring.max(height,
207:                 getConstraintsForCell(r, c, parent, cols).
208:                 getHeight());
209:         }
210:         for (int c = 0; c < cols; c++) {
211:             SpringLayout.Constraints constraints =
212:                 getConstraintsForCell(r, c, parent, cols);
213:             constraints.setY(y);
214:             constraints.setHeight(height);
215:         }
216:         y = Spring.sum(y, Spring.sum(height, Spring.constant(yPad)));
217:     }
218:
219:     //Set the parent's size.
220:     SpringLayout.Constraints pCons = layout.getConstraints(parent);
221:     pCons.setConstraint(SpringLayout.SOUTH, y);
222:     pCons.setConstraint(SpringLayout.EAST, x);
223: }
224: }
225:
```

Table of Contents

1	BIROBox/src/eu/biro/birobox/configuration/BIROAdaptorConfigurationList.java	4 pages	167 lines	09/07/15 12:36:24
2	BIROBox/src/eu/biro/birobox/configuration/BIROBoxConfiguration.java	3 pages	128 lines	09/07/15 12:36:44
3	BIROBox/src/eu/biro/birobox/configuration/BIRODatabaseManagerConfiguration.java	3 pages	135 lines	09/07/15 12:37:20
4	BIROBox/src/eu/biro/birobox/configuration/CommunicationSoftwareConfiguration.java	4 pages	145 lines	09/07/11 15:16:08
5	BIROBox/src/eu/biro/birobox/configuration/Configuration.java	2 pages	45 lines	09/07/15 12:37:46
6	BIROBox/src/eu/biro/birobox/configuration/StatisticalEngineConfiguration.java	4 pages	170 lines	09/07/11 15:16:10
7	BIROBox/src/eu/biro/birobox/main/BIROBoxMain.java	8 pages	309 lines	09/07/15 12:40:00
8	BIROBox/src/eu/biro/birobox/models/BIROFieldTableCellRenderer.java	2 pages	68 lines	09/07/11 15:16:04
9	BIROBox/src/eu/biro/birobox/models/BIROFieldTableModel.java	2 pages	81 lines	09/07/11 15:16:04
10	BIROBox/src/eu/biro/birobox/models/ConfigurationListModel.java	2 pages	69 lines	09/07/11 15:16:04
11	BIROBox/src/eu/biro/birobox/models/DBMSDriverComboBoxModel.java	2 pages	60 lines	09/07/11 15:16:04
12	BIROBox/src/eu/biro/birobox/models/StaticFieldTableCellRenderer.java	2 pages	69 lines	09/07/15 12:41:22
13	BIROBox/src/eu/biro/birobox/models/StaticFieldTableModel.java	3 pages	108 lines	09/07/15 12:41:52
14	BIROBox/src/eu/biro/birobox/models/StatisticalObjectTableCellEditor.java	2 pages	66 lines	09/07/11 15:16:04
15	BIROBox/src/eu/biro/birobox/models/StatisticalObjectTableCellRenderer.java	2 pages	71 lines	09/07/15 12:42:30
16	BIROBox/src/eu/biro/birobox/models/StatisticalObjectTableModel.java	2 pages	88 lines	09/07/15 12:43:00
17	BIROBox/src/eu/biro/birobox/panel/ChildrenPanel.java	1 pages	38 lines	09/07/11 15:16:08
18	BIROBox/src/eu/biro/birobox/panel/HomePanel.java	2 pages	83 lines	09/07/11 15:16:08
19	BIROBox/src/eu/biro/birobox/panel/RootPanel.java	2 pages	59 lines	09/07/15 12:43:44
20	BIROBox/src/eu/biro/birobox/panel/adaptor/ActivityTableConfigurationPanel.java	8 pages	324 lines	09/07/15 12:44:38
21	BIROBox/src/eu/biro/birobox/panel/adaptor/AdaptorProgressPanel.java	4 pages	158 lines	09/07/11 15:16:06
22	BIROBox/src/eu/biro/birobox/panel/adaptor/ConfigurationManagerPanel.java	11 pages	426 lines	09/07/15 12:46:10
23	BIROBox/src/eu/biro/birobox/panel/adaptor/ConnectionConfigurationPanel.java	17 pages	652 lines	09/07/11 15:16:06
24	BIROBox/src/eu/biro/birobox/panel/adaptor/DataSourceConfigurationPanel.java	7 pages	289 lines	09/07/11 15:16:06
25	BIROBox/src/eu/biro/birobox/panel/adaptor/DatePanel.java	3 pages	91 lines	09/07/11 15:16:06
26	BIROBox/src/eu/biro/birobox/panel/adaptor/EnumeratedFieldMappingPanel.java	5 pages	215 lines	09/07/15 12:47:50
27	BIROBox/src/eu/biro/birobox/panel/adaptor/FieldMappingPanel.java	11 pages	444 lines	09/07/15 12:49:04
28	BIROBox/src/eu/biro/birobox/panel/adaptor/FieldPanel.java	2 pages	49 lines	09/07/11 15:16:06
29	BIROBox/src/eu/biro/birobox/panel/adaptor/MergeTableConfigurationPanel.java	12 pages	467 lines	09/07/15 12:50:10
30	BIROBox/src/eu/biro/birobox/panel/adaptor/SaveXMLFilePanel.java	6 pages	246 lines	09/07/15 12:50:32
31	BIROBox/src/eu/biro/birobox/panel/adaptor/UnitOfMeasurementPanel.java	3 pages	98 lines	09/07/11 15:16:06
32	BIROBox/src/eu/biro/birobox/panel/communicationSoftware/CommunicationSoftwarePanel.java	10 pages	405 lines	09/07/15 12:53:54
33	BIROBox/src/eu/biro/birobox/panel/communicationSoftware/StatisticalObject.java	3 pages	118 lines	09/07/11 15:16:08
34	BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseConfigurationPanel.java	10 pages	372 lines	09/07/11 15:16:08
35	BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseManagerPanel.java	7 pages	259 lines	09/07/15 12:55:22
36	BIROBox/src/eu/biro/birobox/panel/databaseManager/DatabaseManagerProgressPanel.java	4 pages	157 lines	09/07/11 12:55:52
37	BIROBox/src/eu/biro/birobox/panel/statisticalEngine/MyEngine.java	2 pages	55 lines	09/07/11 15:16:04
38	BIROBox/src/eu/biro/birobox/panel/statisticalEngine/ROutputFrame.java	9 pages	358 lines	09/07/15 12:57:08
39	BIROBox/src/eu/biro/birobox/panel/statisticalEngine/StatisticalEnginePanel.java	14 pages	549 lines	09/07/11 18:07:32

Table of Contents

40	BIROBox/src/eu/biro/birobox/utils/ConnectionManager.java	3 pages	132 lines	09/07/11 15:16:08
41	BIROBox/src/eu/biro/birobox/utils/CustomDBMSDriver.java	4 pages	144 lines	09/07/15 13:17:16
42	BIROBox/src/eu/biro/birobox/utils/FileListener.java	2 pages	49 lines	09/07/15 13:17:36
43	BIROBox/src/eu/biro/birobox/utils/FileMonitor.java	5 pages	195 lines	09/07/15 13:19:08
44	BIROBox/src/eu/biro/birobox/utils/JarFileLoader.java	2 pages	83 lines	09/07/11 15:32:28
45	BIROBox/src/eu/biro/birobox/utils/JarFinder.java	2 pages	73 lines	09/07/11 15:16:08
46	BIROBox/src/eu/biro/birobox/utils/SpringUtilities.java	5 pages	224 lines	09/07/11 15:16:08