

OSTask.java - svn.adempiere.base.src.org.compiere.util.Task.java

```

1/*****
2 * Copyright (C) 2008 Low Heng Sin *
3 * This program is free software; you can redistribute it and/or modify it *
4 * under the terms version 2 of the GNU General Public License as published *
5 * by the Free Software Foundation. This program is distributed in the hope *
6 * that it will be useful, but WITHOUT ANY WARRANTY; without even the implied *
7 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. *
8 * See the GNU General Public License for more details. *
9 * You should have received a copy of the GNU General Public License along *
10 * with this program; if not, write to the Free Software Foundation, Inc., *
11 * 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. *
12 *****/
13package org.adempiere.webui.util;
14
15import java.io.IOException;
16import java.io.InputStream;
17import java.io.OutputStream;
18import java.util.logging.Level;
19
20import org.compiere.util.CLogger;
21
22/**
23 * Execute OS Task
24 *
25 * @author Low Heng Sin
26 */
27public class OSTask extends Thread
28{
29    /**
30     * Create Process with cmd
31     * @param cmd o/s command
32     */
33    public OSTask (String cmd)
34    {
35        m_cmd = cmd;
36    } // Task
37
38    private String      m_cmd;
39    private Process     m_child = null;
40
41    private StringBuffer m_out = new StringBuffer();
42    private StringBuffer m_err = new StringBuffer();
43
44    /** The Output Stream of process */
45    private InputStream m_outStream;
46    /** The Error Output Stream of process */
47    private InputStream m_errStream;
48    /** The Input Stream of process */
49    private OutputStream m_inStream;
50
51    /** Logger */

```

```

1/*****
2 * Product: Adempiere ERP & CRM Smart Business Solution *
3 * Copyright (C) 1999-2006 ComPiere, Inc. All Rights Reserved. *
4 * This program is free software; you can redistribute it and/or modify it *
5 * under the terms version 2 of the GNU General Public License as published *
6 * by the Free Software Foundation. This program is distributed in the hope *
7 * that it will be useful, but WITHOUT ANY WARRANTY; without even the implied *
8 * warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. *
9 * See the GNU General Public License for more details. *
10 * You should have received a copy of the GNU General Public License along *
11 * with this program; if not, write to the Free Software Foundation, Inc., *
12 * 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA. *
13 * For the text or an alternative of this public license, you may reach us *
14 * ComPiere, Inc., 2620 Augustine Dr. #245, Santa Clara, CA 95054, USA *
15 * or via info@compiere.org or http://www.compiere.org/license.html *
16 *****/
17package org.compiere.util;
18
19import java.io.IOException;
20import java.io.InputStream;
21import java.io.OutputStream;
22import java.util.logging.Level;
23
24/**
25 * Execute OS Task
26 *
27 * @author Jorg Janke
28 * @version $Id: Task.java,v 1.2 2006/07/30 00:51:05 jjanke Exp $
29 */
30public class Task extends Thread
31{
32    /**
33     * Create Process with cmd
34     * @param cmd o/s command
35     */
36    public Task (String cmd)
37    {
38        m_cmd = cmd;
39    } // Task
40
41    private String      m_cmd;
42    private Process     m_child = null;
43
44    private StringBuffer m_out = new StringBuffer();
45    private StringBuffer m_err = new StringBuffer();
46
47    /** The Output Stream of process */
48    private InputStream m_outStream;
49    /** The Error Output Stream of process */
50    private InputStream m_errStream;
51    /** The Input Stream of process */
52    private OutputStream m_inStream;
53
54    /** Logger */

```

```

52 private static CLogger log = CLogger.getLogger(OSTask.class);
53
54 /** Read Out */
55 private Thread m_outReader = new Thread()
56 {
57     public void run()
58     {
59         log.fine("outReader");
60         try
61         {
62             int c;
63             while ((c = m_outStream.read()) != -1 && !isInterrupted())
64             {
65                 m_out.append((char)c);
66             }
67             m_outStream.close();
68         }
69         catch (IOException ioe)
70         {
71             log.log(Level.SEVERE, "outReader", ioe);
72         }
73         log.fine("outReader - done");
74     } // run
75 }; // m_outReader
76
77 /** Read Out */
78 private Thread m_errReader = new Thread()
79 {
80     public void run()
81     {
82         log.fine("errReader");
83         try
84         {
85             int c;
86             while ((c = m_errStream.read()) != -1 && !isInterrupted())
87             {
88                 m_err.append((char)c);
89             }
90             m_errStream.close();
91         }
92         catch (IOException ioe)
93         {
94             log.log(Level.SEVERE, "errReader", ioe);
95         }
96         log.fine("errReader - done");
97     } // run
98 }; // m_errReader
99
100 /**
101  * Execute it
102  */
103
104 public void run()
105 {

```

```

55 private static CLogger log = CLogger.getLogger(Task.class);
56
57 /** Read Out */
58 private Thread m_outReader = new Thread()
59 {
60     public void run()
61     {
62         log.fine("outReader");
63         try
64         {
65             int c;
66             while ((c = m_outStream.read()) != -1 && !isInterrupted())
67             {
68                 // System.out.print((char)c);
69                 m_out.append((char)c);
70             }
71             m_outStream.close();
72         }
73         catch (IOException ioe)
74         {
75             log.log(Level.SEVERE, "outReader", ioe);
76         }
77         log.fine("outReader - done");
78     } // run
79 }; // m_outReader
80
81 /** Read Out */
82 private Thread m_errReader = new Thread()
83 {
84     public void run()
85     {
86         log.fine("errReader");
87         try
88         {
89             int c;
90             while ((c = m_errStream.read()) != -1 && !isInterrupted())
91             {
92                 // System.err.print((char)c);
93                 m_err.append((char)c);
94             }
95             m_errStream.close();
96         }
97         catch (IOException ioe)
98         {
99             log.log(Level.SEVERE, "errReader", ioe);
100         }
101         log.fine("errReader - done");
102     } // run
103 }; // m_errReader
104
105 /**
106  * Execute it
107  */
108
109 public void run()
110 {

```

```

106 log.info(m_cmd);
107 try
108 {
109     m_child = Runtime.getRuntime().exec(m_cmd);
110     //
111     m_outStream = m_child.getInputStream();
112     m_errStream = m_child.getErrorStream();
113     m_inStream = m_child.getOutputStream();
114     //
115     if (checkInterrupted())
116         return;
117     m_outReader.start();
118     m_errReader.start();
119
120     Integer exitValue = null;
121     while(exitValue == null)
122     {
123         //
124         try
125         {
126             Thread.sleep(500);
127             if (checkInterrupted())
128                 return;
129             int i = m_child.exitValue();
130             exitValue = new Integer(i);
131         }
132         catch (Exception ie)
133         {
134             log.log(Level.INFO, "(ie) - " + ie);
135         }
136         // ExitValue
137         log.config("done");
138     }
139 }
140 catch (IOException ioe)
141 {
142     log.log(Level.SEVERE, "(ioe)", ioe);
143 }
144 // run
145
146 /**
147  * Check if interrupted
148  * @return true if interrupted
149  */
150 private boolean checkInterrupted()
151 {
152     if (isInterrupted())
153     {
154         log.config("interrupted");
155         // interrupt child processes

```

```

111 log.info(m_cmd);
112 try
113 {
114     m_child = Runtime.getRuntime().exec(m_cmd);
115     //
116     m_outStream = m_child.getInputStream();
117     m_errStream = m_child.getErrorStream();
118     m_inStream = m_child.getOutputStream();
119     //
120     if (checkInterrupted())
121         return;
122     m_outReader.start();
123     m_errReader.start();
124     //
125     try
126     {
127         if (checkInterrupted())
128             return;
129         m_errReader.join();
130         if (checkInterrupted())
131             return;
132         m_outReader.join();
133         if (checkInterrupted())
134             return;
135         m_child.waitFor();
136     }
137     catch (InterruptedException ie)
138     {
139         log.log(Level.INFO, "(ie) - " + ie);
140     }
141     // ExitValue
142     try
143     {
144         if (m_child != null)
145             log.fine("run - ExitValue=" + m_child.exitValue());
146     }
147     catch (Exception e) {}
148     log.config("done");
149 }
150 catch (IOException ioe)
151 {
152     log.log(Level.SEVERE, "(ioe)", ioe);
153 }
154 // run
155
156 /**
157  * Check if interrupted
158  * @return true if interrupted
159  */
160 private boolean checkInterrupted()
161 {
162     if (isInterrupted())
163     {
164         log.config("interrupted");
165         // interrupt child processes

```

OSTask.java - svn.adempiere.base.src.org.compiere.util.Task.java

```

156     if (m_child != null)
157         m_child.destroy();
158     m_child = null;
159     if (m_outReader != null && m_outReader.isAlive())
160         m_outReader.interrupt();
161     m_outReader = null;
162     if (m_errReader != null && m_errReader.isAlive())
163         m_errReader.interrupt();
164     m_errReader = null;
165     // close Streams
166     if (m_inStream != null)
167         try { m_inStream.close(); } catch (Exception e) {}
168     m_inStream = null;
169     if (m_outStream != null)
170         try { m_outStream.close(); } catch (Exception e) {}
171     m_outStream = null;
172     if (m_errStream != null)
173         try { m_errStream.close(); } catch (Exception e) {}
174     m_errStream = null;
175     //
176     return true;
177 }
178 return false;
179 } // checkInterrupted
180
181 /**
182  * Get Out Info
183  * @return StringBuffer
184  */
185 public StringBuffer getOut()
186 {
187     return m_out;
188 } // getOut
189
190 /**
191  * Get Err Info
192  * @return StringBuffer
193  */
194 public StringBuffer getErr()
195 {
196     return m_err;
197 } // getErr
198
199 /**
200  * Get The process input stream - i.e. we output to it
201  * @return OutputStream
202  */
203 public OutputStream getInStream()
204 {
205     return m_inStream;
206 } // getInStream
207
208 // Task
209

```

```

166     if (m_child != null)
167         m_child.destroy();
168     m_child = null;
169     if (m_outReader != null && m_outReader.isAlive())
170         m_outReader.interrupt();
171     m_outReader = null;
172     if (m_errReader != null && m_errReader.isAlive())
173         m_errReader.interrupt();
174     m_errReader = null;
175     // close Streams
176     if (m_inStream != null)
177         try { m_inStream.close(); } catch (Exception e) {}
178     m_inStream = null;
179     if (m_outStream != null)
180         try { m_outStream.close(); } catch (Exception e) {}
181     m_outStream = null;
182     if (m_errStream != null)
183         try { m_errStream.close(); } catch (Exception e) {}
184     m_errStream = null;
185     //
186     return true;
187 }
188 return false;
189 } // checkInterrupted
190
191 /**
192  * Get Out Info
193  * @return StringBuffer
194  */
195 public StringBuffer getOut()
196 {
197     return m_out;
198 } // getOut
199
200 /**
201  * Get Err Info
202  * @return StringBuffer
203  */
204 public StringBuffer getErr()
205 {
206     return m_err;
207 } // getErr
208
209 /**
210  * Get The process input stream - i.e. we output to it
211  * @return OutputStream
212  */
213 public OutputStream getInStream()
214 {
215     return m_inStream;
216 } // getInStream
217
218 // Task
219

```