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

```
/******************************************************************************
* Copyright (C) 2008 Low Heng Sin                                             *
* This program is free software; you can redistribute it and/or modify it     *
* under the terms version 2 of the GNU General Public License as published    *
* by the Free Software Foundation. This program is distributed in the hope    *
* that it will be useful, but WITHOUT ANY WARRANTY; without even the implied  *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.            *
* See the GNU General Public License for more details.                        *
* You should have received a copy of the GNU General Public License along     *
* with this program; if not, write to the Free Software Foundation, Inc.,     *
* 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.                      *
*****************************************************************************/
package org.adempiere.webui.util;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.logging.Level;

import org.compiere.util.CLogger;

/**
 *  Execute OS Task
 *
 *  @author Low Heng Sin

 */
public class OSTask extends Thread
{
    /**
     *  Create Process with cmd
     *  @param cmd o/s command
     */
    public OSTask (String cmd)
    {
        m_cmd = cmd;
    }   //  Task

    private String          m_cmd;
    private Process         m_child = null;

    private StringBuffer    m_out = new StringBuffer();
    private StringBuffer    m_err = new StringBuffer();

    /** The Output Stream of process        */
    private InputStream     m_outStream;
    /** The Error Output Stream of process  */
    private InputStream     m_errStream;
    /** The Input Stream of process         */
    private OutputStream    m_inStream;

    /** Logger          */
```

```
/******************************************************************************
* Product: Adempiere ERP & CRM Smart Business Solution                        *
* Copyright (C) 1999-2006 ComPiere, Inc. All Rights Reserved.                 *
* This program is free software; you can redistribute it and/or modify it     *
* under the terms version 2 of the GNU General Public License as published    *
* by the Free Software Foundation. This program is distributed in the hope    *
* that it will be useful, but WITHOUT ANY WARRANTY; without even the implied  *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.            *
* See the GNU General Public License for more details.                        *
* You should have received a copy of the GNU General Public License along     *
* with this program; if not, write to the Free Software Foundation, Inc.,     *
* 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA.                      *
* For the text or an alternative of this public license, you may reach us     *
* ComPiere, Inc., 2620 Augustine Dr. #245, Santa Clara, CA 95054, USA         *
* or via info@compiere.org or http://www.compiere.org/license.html            *
*****************************************************************************/
package org.compiere.util;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.logging.Level;



/**
 *  Execute OS Task
 *
 *  @author     Jorg Janke
 *  @version    $Id: Task.java,v 1.2 2006/07/30 00:51:05 jjanke Exp $
 */
public class Task extends Thread
{
    /**
     *  Create Process with cmd
     *  @param cmd o/s command
     */
    public Task (String cmd)
    {
        m_cmd = cmd;
    }   //  Task

    private String          m_cmd;
    private Process         m_child = null;

    private StringBuffer    m_out = new StringBuffer();
    private StringBuffer    m_err = new StringBuffer();

    /** The Output Stream of process        */
    private InputStream     m_outStream;
    /** The Error Output Stream of process  */
    private InputStream     m_errStream;
    /** The Input Stream of process         */
    private OutputStream    m_inStream;

    /** Logger          */
```

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

```
52    private static CLogger log = CLogger.getCLogger(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   /**
102    *  Execute it
103    */
104   public void run()
105   {
```

```
55    private static CLogger log = CLogger.getCLogger(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   /**
107    *  Execute it
108    */
109   public void run()
110   {
```

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

```
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

```
            if (m_child != null)
                m_child.destroy();
            m_child = null;
            if (m_outReader != null && m_outReader.isAlive())
                m_outReader.interrupt();
            m_outReader = null;
            if (m_errReader != null && m_errReader.isAlive())
                m_errReader.interrupt();
            m_errReader = null;
            //  close Streams
            if (m_inStream != null)
                try {   m_inStream.close();     } catch (Exception e) {}
            m_inStream = null;
            if (m_outStream != null)
                try {   m_outStream.close();    } catch (Exception e) {}
            m_outStream = null;
            if (m_errStream != null)
                try {   m_errStream.close();    } catch (Exception e) {}
            m_errStream = null;
            //
            return true;
        }
        return false;
    }   //  checkInterrupted

    /**
     *  Get Out Info
     *  @return StringBuffer
     */
    public StringBuffer getOut()
    {
        return m_out;
    }   //  getOut

    /**
     *  Get Err Info
     *  @return StringBuffer
     */
    public StringBuffer getErr()
    {
        return m_err;
    }   //  getErr

    /**
     *  Get The process input stream - i.e. we output to it
     *  @return OutputStream
     */
    public OutputStream getInStream()
    {
        return m_inStream;
    }   //  getInStream

}   //  Task
```