```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package cmseditor;

import java.awt.Color;
import java.awt.*;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Rectangle;
import java.awt.event.MouseEvent;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.tree.DefaultMutableTreeNode;

/**
 *
 * @author Rod
 *
 * This was originally a class defined WITHIN PictureSort, but needing it too for the new PictureIndex
 * I moved it into a separate definition as here
 *
 * BE VERY CAREFUL about the way Java always passes by VALUE, so although the arrays are echoed back into the calling
environment,
 * any changes to ImageCount ARE NOT!!
 *
 */
public class CMSImagePanel  extends JPanel
{
   /*
      BufferedImage [] images;
     DefaultMutableTreeNode [] nodes;
     Rectangle [] rects;
     int [] ix;*/
     ASLArray<PanelImage> Images = new ASLArray(50, 50, PanelImage.class);
     ASLArray<Integer> Index = new ASLArray(50, 50, int.class);
     public int rowoff, coloff, height, DragImageIx = -1, DragDestIx = -1, LastMouseIx = -1;
     CMSUtilities UTILS;
     NewEditorCode NE = null;
     JFrame CallingForm;
     String Breadcrumb;
     MouseEvent lastmouseevent;
     //boolean WasPopup = false;

     public CMSImagePanel( int hght, int imgCount, CMSUtilities utils)
     {
        //images = imgs; nodes = nds; rects = rcts; ix = iix;
        height = hght;
        Images.count = imgCount;
        UTILS = utils;
        //CallingForm = caller;
        //Breadcrumb = brdcrmb;
         java.awt.event.MouseListener ml = new java.awt.event.MouseAdapter()
```

```
    {
        public void mousePressed(java.awt.event.MouseEvent e)
        {
            StartDrag(e);  return;
        }
        public void mouseReleased(MouseEvent e)
        {
            OnDrop(e); return;
        }
        public void mouseDragged(MouseEvent e)
        {
            WhileDrag(e); return;
        }
        public void mouseClicked(MouseEvent e)
        {
            MouseClicked(e);  return;
        }
    };
   // tree.addMouseListener(ml);
   this.addMouseListener(ml);

}

  @Override
  public void paint(Graphics g)
  {
     rowoff = 10; coloff = 10;
     float ar;
     //g.drawImage(img1, 0, 0, this);
     Font fnt = g.getFont();
     fnt.deriveFont(Font.PLAIN, 8);
     //Rectangle rr = new Rectangle();
     //this.setBackground(Color.BLACK);
     g.setColor(Color.BLACK);
     g.fillRect(0, 0, this.getWidth(), this.getHeight());
     for (int ii = 1; ii <= Images.count; ii++)
     {
        //int jj = ix(ii);
        BufferedImage tmpimg = img(ii);
        if (tmpimg != null)
        {
            ar = (float)tmpimg.getWidth()/(float)tmpimg.getHeight();
            g.drawImage(tmpimg, coloff, rowoff, (int)(ar * height), height, this);
        }
        else { ar = 1; }
        Images.item(ix(ii)).rect = new Rectangle(coloff, rowoff, (int)(ar * height), height);
        if (tmpimg == null)
        {
            g.setColor(Color.DARK_GRAY);
            g.fillRect(coloff, rowoff, height,height);
        }
        DefaultMutableTreeNode tn = node(ii);
        if (tn == null) continue;
        ASLXMLNode xn = UTILS.GetXMLNode( tn); //Images.Elems[jj].node);
        if (xn != null)
        {
            String caption;
            if (xn.tagname.equalsIgnoreCase("SaleItem")) caption = xn.GetAttributeValue("ItemName");
```

```
            else caption = xn.GetAttributeValue("Caption");
            if (!caption.isEmpty())
            {
               g.setFont(fnt);
               g.setColor(Color.yellow);
               g.drawString(caption, coloff, rowoff + height + 12); //these aligned along the bottom of the text
               g.setColor(Color.black);
               g.fillRect(coloff + (int)(ar * height), rowoff + height, 100, 20); //truncate it to picture width
            }
            else
            {
               caption = xn.value;
               g.setFont(fnt);
               g.setColor(Color.LIGHT_GRAY);
               g.drawString(caption, coloff, rowoff + height + 12); //these aligned along the bottom of the text
               g.setColor(Color.black);
               g.fillRect(coloff + (int)(ar * height), rowoff + height, 300, 20); //truncate it to picture width
            }
         }
         coloff += ar * height + 20;
         if (coloff + 120 > this.getWidth())
         {
            coloff = 10; rowoff += height + 30;
         }
      }
      if (rowoff > this.getHeight())
      {
         this.setPreferredSize(new Dimension(this.getWidth(), rowoff + 50));
        //this.repaint();
         this.getParent().invalidate();
         //nav.invalidate();
         //nav.repaint();
      }
     this.setCursor(Cursor.getDefaultCursor());
  } //end paint

   public class myInt
   {
      public int ix;
      public myInt(int ii) {ix = ii;}
      //curiously, the built-in Integer class kicks here
   }
   public int ix(int ii)
   {
      if (ii < 1 || ii > Index.count) return 0;
      return Index.item(ii);
   } // Images.item(ii).ix; }
   public BufferedImage img(int ii)
   {
      int jj = ix(ii); if (jj == 0) return null;
      return Images.item(jj).image;
   }
   public DefaultMutableTreeNode node(int ii)
   {
      int jj = ix(ii); if (jj == 0) return null;
      return Images.item(jj).node;
   }
   public Rectangle rect(int ii)
```

```
   {
      int jj = ix(ii); if (jj == 0) return null;
      return Images.item(jj).rect;
   }

int AddImage(BufferedImage nimg, DefaultMutableTreeNode nd, Rectangle r)
{
   int ii = Images.add(new PanelImage(Images.count + 1, nd, nimg, r));
   Index.add(ii); //new myInt(ii));
   return ii;
}
int LocateMouseEvent(MouseEvent e)
{
    int ii;
   for ( ii = 1; ii <= Images.count; ii++)
   {
      //int jj = Images.item(ii).ix;
      Rectangle r = rect(ii); // Images.Elems[jj].rect;   // rects[ix[ii]];
      if (r == null) continue;
      int x = e.getX(), y = e.getY();
      if (y >= r.y && y < r.y + r.height)
      {
         if (x >= r.x && x < r.x + r.width) return ii;
      }
   }
   return -1;
}
void MouseClicked(java.awt.event.MouseEvent e)
{
   //if (WasPopup)
   //{
      //DragImageIx = LocateMouseEvent(e);
    //   return;
   //}
   DragImageIx = -1; //clear that - not now drag and drop

   this.setCursor(Cursor.getDefaultCursor());
   int ii = LocateMouseEvent(e);
   if (ii == 0) return;
   if (e.getClickCount() == 2)
   {
      //int jj = Images.Elems[ii].ix;
      ASLXMLNode xn;
      DefaultMutableTreeNode tn = node(ii);
      if (tn == null) return;
      try{ xn= UTILS.GetXMLNode(tn); }// Images.Elems[jj].node);  //nodes[ix[ii]]);
      catch(Exception ee)
      {
         JOptionPane.showMessageDialog(this, "Error - Missed the box!  - "+ ee.getMessage());
         return;
      }
      if (xn.tagname.equalsIgnoreCase("saleitem"))
      {
         NE.EditSaleItem(this.CallingForm, tn, Breadcrumb);
         return;
      }
      if (xn.GetAttributeValue("Type").equalsIgnoreCase("HTML"))
      {
```

```
                File fil = new File(UTILS.GetFullPath(tn)); //Oct19-  XND.GetAttributeValue("LocalFilePath"));
                String url = fil.toURI().toString();
                NE.EditHTML(fil, url);
                return;
            }
            if (!tn.isLeaf() || xn.tagname.equalsIgnoreCase("Folder"))
            {
                int piccnt = 0, foldercnt = 0;
                boolean gofolder = false, isAmbig = false;
                String ambigmsg = "";
                //if (tnd.isLeaf()) return;
                if (tn.getChildCount() > 0)
                {
                    DefaultMutableTreeNode tnd1 = (DefaultMutableTreeNode) tn.getFirstChild();
                    if (tnd1 == null) return;
                    while (tnd1 != null)
                    {
                        ASLXMLNode xnd = UTILS.GetXMLNode(tnd1);
                        if (xnd != null)
                        {
                            if (xnd.GetAttributeValue("Type").equalsIgnoreCase("Image")) piccnt++;
                            if (xnd.tagname.equalsIgnoreCase("folder"))
                            {
                                if (!xnd.value.equalsIgnoreCase("textimages") && !xnd.value.equalsIgnoreCase("reduced") &&
!xnd.value.equalsIgnoreCase("thumbnails")) foldercnt++;
                            }
                            if (xnd.tagname.equalsIgnoreCase("saleitem")) foldercnt++;
                        }
                        tnd1 = tnd1.getNextSibling();
                    }
                }
                if (foldercnt == 0 && piccnt == 0)
                {
                    isAmbig = true; ambigmsg = "This Folder has neither Images nor Subfolders\n\n";
                }
                else if (foldercnt > 0 && piccnt > 1)  //allow one index image
                {
                    isAmbig = true; ambigmsg = "This Folder contains BOTH Images and Sub-Folders\n\n";
                }
                if (foldercnt > 0) gofolder = true;
                String typ = xn.GetAttributeValue("FolderType");
                if (typ.length() > 0)
                    if (typ.equalsIgnoreCase("Project")) { gofolder = false; isAmbig = false;} else { gofolder = true; isAmbig = false;}
                if (isAmbig)
                {
                    int yn = JOptionPane.showConfirmDialog(this, ambigmsg+ "Open as simple Images Folder (rather than as
category of folders)?");
                    if (yn == JOptionPane.CANCEL_OPTION) return;
                    gofolder = false;
                    if (yn == JOptionPane.NO_OPTION) gofolder = true;
                    yn = JOptionPane.showConfirmDialog(this, "Would you like this folder marked as this type hereafter?");
                    if (yn == JOptionPane.OK_OPTION)
                    {
                        //ASLXMLNode xnd = UTILS.GetXMLNode(tnd);
                        if (gofolder) xn.AddAttribute("FolderType", "Category"); else xn.AddAttribute("FolderType", "Project");
                    }
                }
            // this.CallingForm.setCursor(Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR));
```

```
            //WaitMessageBox WM = new WaitMessageBox(null);
            //WM.setLocationRelativeTo(null);
            //WM.pack();
            // NE.Wait4Me.setLocationRelativeTo(this.CallingForm);
                NE.Wait4Me.setVisible(true);
            NE.Wait4Me.repaint();
            if (!gofolder)  //was piccnt > 1)
            {
              this.CallingForm.setVisible(false);
              PictureSort PS = new PictureSort(this.CallingForm, NE, UTILS, Breadcrumb,  NE.NE.XMLTree, tn,
NE.NE.txtPanel, 90, null);
              PS.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
              PS.setLocationRelativeTo(this.CallingForm);
              PS.pack();
              //PS.setModal(true);
              PS.setVisible(true);

            }
            else
            {
              this.CallingForm.setVisible(false);
               PictureIndex PI = new PictureIndex(this.CallingForm, NE, UTILS, Breadcrumb, NE.NE.XMLTree, tn,
NE.NE.txtPanel, 90);
              PI.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
              PI.setLocationRelativeTo(this.CallingForm);
              PI.pack();
              //PS.setModal(true);
              PI.setVisible(true);
            }
            NE.Wait4Me.setVisible(false);              //WM.dispose();
          }
          else
          {
             String caption = xn.GetAttributeValue("Caption");
             if (caption.isEmpty()) caption = xn.value;
             caption = JOptionPane.showInputDialog(this.getParent(),  "Caption", caption);
             int yesno = -1;
             if (caption == null)  {   yesno = 0;  caption = ""; }  else if (caption.length() == 0)    { yesno = 0;   }
             if (yesno == 0)
             {
                     Object[] options = {"Yes, I do","No, I don't"  };
                      yesno = JOptionPane.showOptionDialog(this, "Are you sure you want to erase this Caption?",
                                 "Caption",    JOptionPane.YES_NO_OPTION,
                                    JOptionPane.QUESTION_MESSAGE,
                                    null,
                                    options,
                                     options[1]);
                      if (yesno == JOptionPane.NO_OPTION) {this.repaint(); return;}
             }
             xn.setAttributeValue("Caption", caption);
          }
          this.repaint();
          return;
       }
       this.repaint();
    }
    void StartDrag(MouseEvent e)
    {
```

```
    lastmouseevent = e;
    LastMouseIx = LocateMouseEvent(e);
    DragImageIx = LastMouseIx; //'ocateMouseEvent(e);
    this.setCursor(Cursor.getPredefinedCursor(Cursor.MOVE_CURSOR));
    int ii = DragImageIx;
    if (ii >= 0)
    {
       //int jj = Images.item(ii).ix;
       Rectangle rect = rect(ii); //Images.Elems[jj].rect; //rects[ix[ii]];
       if (rect == null) return;
       Graphics g = getGraphics();
       g.setColor(Color.green);
       g.drawRect( rect.x - 1, rect.y - 1, rect.width + 2, rect.height + 2);
       g.setColor(Color.black);
       //pane.repaint();
    }
}
void WhileDrag(MouseEvent e)
{
    //These events are never returned in Windows, so since can't debug, abandon
    //would need to "unpaint" the rectangle from the last DragDestIx
    //if (DragImageIx < 0) StartDrag(e);
    if (DragImageIx < 0) return;
    int ii = LocateMouseEvent(e);
    if (ii > 0 && ii <= Images.count && ii != DragImageIx && ii != DragDestIx)
    {
       DragDestIx = ii;
    }
}
int getix(int ii) {return ix(ii); } // Images.Elems[ii].ix; }
void setix(int ii, int jj)
{
    Index.set(ii, jj);  //for some odd reason have to do this assigment WITHIN the ASLArray class
    Images.item(jj).ix = ii;
}

void OnDrop(MouseEvent e)
{
    LastMouseIx = LocateMouseEvent(e);
    if (DragImageIx < 0) return;
    int ii = LocateMouseEvent(e);

    if (ii >= 0 && ii != DragImageIx)
    {
       if (ii < DragImageIx)
       {
          int k = ix(DragImageIx); //for (int jj = DragImageIx; jj > ii; jj--) ix[jj] = ix[jj-1];
          for (int jj = DragImageIx; jj > ii; jj--) setix(jj, ix(jj - 1));
          //ix[ii] = k; //sortpnt++;
          setix(ii, k);
       }
       else
       {
          //int k = ix[DragImageIx]; for (int jj = DragImageIx; jj < ii; jj++) ix[jj] = ix[jj+1];
          //ix[ii] = k;
          int k = ix(DragImageIx);
          for (int jj = DragImageIx; jj < ii; jj++) setix(jj, ix(jj + 1));
          setix(ii, k);
```

```
        }
        this.repaint();
      }
      else    if (DragImageIx >= 0)
      {
        // int jj = Images.item(ii).ix;
        Rectangle rect = rect(DragImageIx);
        if (rect == null) return;
        Graphics g = getGraphics();
        g.setColor(Color.black);
        g.drawRect(rect.x - 1, rect.y - 1, rect.width + 2, rect.height + 2);
        //wipe the source marker rectangle
      }

      this.setCursor(Cursor.getDefaultCursor());
      DragImageIx = DragDestIx = -1;
    }

    void Delete(int k)
    {
      int k1 = ix(k); //cos this will go
      Images.remove(ix(k));
      Index.remove(k);
      for (int ii = 1; ii <= Index.count; ii++) if (ix(ii) > k1) setix(ii, ix(ii) - 1);
      return;
    }

}
```