com.fr.design.fun.BackgroundQuickUIProvider

-
-
-
-
-
-
-
-
-

10.0“”10.08.0“”“”

BackgroundQuickUIProvider.java

```
package com.fr.design.fun;

import com.fr.design.mainframe.backgroundpane.BackgroundQuickPane;
import com.fr.stable.fun.Level;
import com.fr.stable.fun.Provider;
import com.fr.stable.fun.mark.Mutable;

/**
 * Created by richie on 16/5/18.
 * ,
 */
public interface BackgroundQuickUIProvider extends Mutable {

    String MARK_STRING = "BackgroundQuickUIProvider";

    int CURRENT_LEVEL = 1;

    /**
     *
     * @return
     */
    BackgroundQuickPane appearanceForBackground();
}
```

ColorDetailPane.java

```
package com.fr.design.mainframe.predefined.ui.detail.background;

import com.fr.base.background.ColorBackground;
import com.fr.design.event.UIObserver;
import com.fr.design.event.UIObserverListener;
import com.fr.design.gui.ibutton.UIButton;
import com.fr.design.gui.ilable.UILabel;
import com.fr.design.layout.FRGUIPaneFactory;
import com.fr.design.layout.TableLayoutHelper;
import com.fr.design.style.color.ColorSelectPane;
import com.fr.general.Background;

import javax.swing.BorderFactory;
```

```
import javax.swing.JPanel;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;

/**
 * Created by kerry on 2020-08-31
 */
public class ColorDetailPane extends AbstractBackgroundDetailPane<ColorBackground> {
    private ColorBackgroundSelectPane selectPane;


    public ColorDetailPane() {
        this.selectPane = new ColorBackgroundSelectPane();
        this.setLayout(FRGUIPaneFactory.createBorderLayout());
        this.add(this.selectPane, BorderLayout.CENTER);
    }

    @Override
    public void populate(ColorBackground background) {
        this.selectPane.setColor(background.getColor());
    }

    @Override
    public ColorBackground update() {
        return ColorBackground.getInstance(selectPane.getColor());
    }

    public String title4PopupWindow() {
        return com.fr.design.i18n.Toolkit.i18nText("Fine-Design_Basic_Color");
    }

    @Override
    public boolean accept(Background background) {
        return background instanceof ColorBackground;
    }

    class ColorBackgroundSelectPane extends ColorSelectPane implements UIObserver {
        protected UIObserverListener uiObserverListener;

        protected void initialCompents(boolean isSupportTransparent) {
            this.setLayout(FRGUIPaneFactory.createBorderLayout());
            this.setBorder(BorderFactory.createEmptyBorder());
            if (isSupportTransparent) {
                this.add(createNorthPane(), BorderLayout.NORTH);
            }
            JPanel centerPane = createCenterPane();
            this.add(centerPane, BorderLayout.CENTER);
            this.addChangeListener(new ChangeListener() {
                @Override
                public void stateChanged(ChangeEvent e) {
                    if (uiObserverListener != null) {
                        uiObserverListener.doChange();
                    }
                }
            });
        }

        private JPanel createNorthPane() {
//            UIButton transpanrentBtn = createTranspanrentButton();
            UIButton transpanrentBtn = new UIButton();
            transpanrentBtn.setPreferredSize(new Dimension(160, 20));
            JPanel jPanel = TableLayoutHelper.createGapTableLayoutPane(
                    new Component[][]{new Component[]{new UILabel(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Background_Color")),
                            transpanrentBtn}}, TableLayoutHelper.FILL_NONE, 33, 5);
            jPanel.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 10));
            return jPanel;
        }
```

```
        protected JPanel createCenterPane() {
//            JPanel centerPane = super.createCenterPane();
            JPanel centerPane = new JPanel();

            JPanel jPanel = TableLayoutHelper.createGapTableLayoutPane(
                    new Component[][]{new Component[]{new UILabel("    "), centerPane}}, TableLayoutHelper.
FILL_NONE, 33, 5);
            jPanel.setBorder(BorderFactory.createEmptyBorder(5, 0, 5, 10));
            return jPanel;
        }

        @Override
        public void registerChangeListener(UIObserverListener listener) {
            this.uiObserverListener = listener;
        }

        @Override
        public boolean shouldResponseChangeListener() {
            return true;
        }
    }
}
```

ImageDetailPane.java

```
package com.fr.design.mainframe.predefined.ui.detail.background;

import com.fr.base.Style;
import com.fr.base.background.ImageBackground;
import com.fr.base.background.ImageFileBackground;
import com.fr.design.designer.IntervalConstants;
import com.fr.design.event.UIObserver;
import com.fr.design.event.UIObserverListener;
import com.fr.design.gui.frpane.ImgChooseWrapper;
import com.fr.design.gui.ibutton.UIButton;
import com.fr.design.gui.ibutton.UIRadioButton;
import com.fr.design.gui.ilable.UILabel;
import com.fr.design.layout.FRGUIPaneFactory;
import com.fr.design.layout.TableLayoutHelper;
import com.fr.design.style.background.image.ImageFileChooser;
import com.fr.design.style.background.image.ImagePreviewPane;
import com.fr.general.Background;
import com.fr.stable.Constants;

import javax.swing.BorderFactory;
import javax.swing.ButtonGroup;
import javax.swing.JPanel;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.BorderLayout;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

/**
 * Image background pane.
 */
public class ImageDetailPane extends AbstractBackgroundDetailPane<ImageBackground> implements UIObserver {
    private UIObserverListener listener;
    protected ImagePreviewPane previewPane = null;
    private Style imageStyle = null;
    private ChangeListener changeListener = null;
    private ImageFileChooser imageFileChooser = null;

    private UIRadioButton defaultRadioButton = null;
```

```
    private UIRadioButton tiledRadioButton = null;
    private UIRadioButton extendRadioButton = null;
    private UIRadioButton adjustRadioButton = null;


    public ImageDetailPane() {
        this.setLayout(FRGUIPaneFactory.createBorderLayout());
        this.add(initSelectFilePane(), BorderLayout.CENTER);
        imageFileChooser = new ImageFileChooser();
        imageFileChooser.setMultiSelectionEnabled(false);
        previewPane = new ImagePreviewPane();
        this.addChangeListener(new ChangeListener() {
            @Override
            public void stateChanged(ChangeEvent e) {
                if (listener != null) {
                    listener.doChange();
                }
            }
        });
    }

    public JPanel initSelectFilePane() {
        JPanel selectFilePane = FRGUIPaneFactory.createBorderLayout_L_Pane();
        selectFilePane.setBorder(BorderFactory.createEmptyBorder());
        UIButton selectPictureButton = new UIButton(
                com.fr.design.i18n.Toolkit.i18nText("Fine-Design_Basic_Background_Image_Select"));
        selectPictureButton.setMnemonic('S');
        selectPictureButton.addActionListener(selectPictureActionListener);
        selectPictureButton.setPreferredSize(new Dimension(160, 20));
        //
        defaultRadioButton = new UIRadioButton(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Style_Alignment_Layout_Default"));
        tiledRadioButton = new UIRadioButton(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Style_Alignment_Layout_Image_Titled"));
        extendRadioButton = new UIRadioButton(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Style_Alignment_Layout_Image_Extend"));
        adjustRadioButton = new UIRadioButton(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Style_Alignment_Layout_Image_Adjust"));

        defaultRadioButton.addActionListener(layoutActionListener);
        tiledRadioButton.addActionListener(layoutActionListener);
        extendRadioButton.addActionListener(layoutActionListener);
        adjustRadioButton.addActionListener(layoutActionListener);

        JPanel jp = new JPanel(new GridLayout(4, 1, 15, 10));
        for (UIRadioButton button : imageLayoutButtons()) {
            jp.add(button);
        }

        ButtonGroup layoutBG = new ButtonGroup();
        layoutBG.add(defaultRadioButton);
        layoutBG.add(tiledRadioButton);
        layoutBG.add(extendRadioButton);
        layoutBG.add(adjustRadioButton);

        defaultRadioButton.setSelected(true);

        Component[][] components = new Component[][]{
                new Component[]{new UILabel(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Background_Image")), selectPictureButton},
                new Component[]{new UILabel(com.fr.design.i18n.Toolkit.i18nText("Fine-
Design_Basic_Background_Fill_Mode")), jp}
        };
        JPanel centerPane = TableLayoutHelper.createGapTableLayoutPane(components, TableLayoutHelper.FILL_NONE,
                IntervalConstants.INTERVAL_L4, IntervalConstants.INTERVAL_L1);
        selectFilePane.add(centerPane, BorderLayout.CENTER);
        return selectFilePane;
    }

    protected UIRadioButton[] imageLayoutButtons() {
        return new UIRadioButton[]{
```

```
                defaultRadioButton,
                tiledRadioButton,
                extendRadioButton,
                adjustRadioButton
        };
    }

    @Override
    public boolean accept(Background background) {
        return background instanceof ImageBackground;
    }


    /**
     * Select picture.
     */
    ActionListener selectPictureActionListener = new ActionListener() {

        public void actionPerformed(ActionEvent evt) {
            int returnVal = imageFileChooser.showOpenDialog(ImageDetailPane.this);
            setImageStyle();
            ImgChooseWrapper.getInstance(previewPane, imageFileChooser, imageStyle, changeListener).
dealWithImageFile(returnVal);
        }
    };

    protected void setImageStyle() {
        if (tiledRadioButton.isSelected()) {
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_TILED);
        } else if (adjustRadioButton.isSelected()) {
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_ADJUST);
        } else if (extendRadioButton.isSelected()) {
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_EXTEND);
        } else {
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_CENTER);
        }
    }

    ActionListener layoutActionListener = new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent evt) {
            setImageStyle();
            changeImageStyle();
        }

        private void changeImageStyle() {
            previewPane.setImageStyle(ImageDetailPane.this.imageStyle);
            previewPane.repaint();
        }
    };

    @Override
    public void populate(ImageBackground imageBackground) {
        if (imageBackground.getLayout() == Constants.IMAGE_CENTER) {
            defaultRadioButton.setSelected(true);
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_CENTER);
        } else if (imageBackground.getLayout() == Constants.IMAGE_EXTEND) {
            extendRadioButton.setSelected(true);
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_EXTEND);
        } else if (imageBackground.getLayout() == Constants.IMAGE_ADJUST) {
            adjustRadioButton.setSelected(true);
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_ADJUST);
        } else {
            tiledRadioButton.setSelected(true);
            imageStyle = Style.DEFAULT_STYLE.deriveImageLayout(Constants.IMAGE_TILED);
        }
        previewPane.setImageStyle(ImageDetailPane.this.imageStyle);
        if (imageBackground.getImage() != null) {
            previewPane.setImageWithSuffix(imageBackground.getImageWithSuffix());
            previewPane.setImage(imageBackground.getImage());
```

```
        }

        fireChagneListener();
    }

    @Override
    public ImageBackground update() {
        ImageBackground imageBackground = new ImageFileBackground(previewPane.getImageWithSuffix());
        setImageStyle();
        imageBackground.setLayout(imageStyle.getImageLayout());
        return imageBackground;
    }

    @Override
    public void addChangeListener(ChangeListener changeListener) {
        this.changeListener = changeListener;
    }

    private void fireChagneListener() {
        if (this.changeListener != null) {
            ChangeEvent evt = new ChangeEvent(this);
            this.changeListener.stateChanged(evt);
        }
    }

    @Override
    public void registerChangeListener(UIObserverListener listener) {
        this.listener = listener;
    }


    @Override
    public String title4PopupWindow() {
        return com.fr.design.i18n.Toolkit.i18nText("Fine-Design_Basic_Background_Image");
    }


}
```

Background.java

```
/*
 * Copyright(c) 2001-2010, FineReport Inc, All Rights Reserved.
 */
package com.fr.general;

import com.fr.common.annotations.Open;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.stable.bridge.ObjectHolder;
import com.fr.stable.web.Repository;
import com.fr.stable.xml.XMLPrintWriter;
import com.fr.stable.xml.XMLableReader;

import java.awt.*;

/**
 *
 * com.fr.base
 */
@Open
public interface Background extends Cloneable, java.io.Serializable {

    /**
     *
     *
     * @param g
```

```
 * @param shape
 */
void paint(Graphics g, Shape shape);

/**
 *
 * @param g
 * @param repo
 * @param shape
 */
void paint(Graphics g, Repository repo, Shape shape);

    /**
     *
     * @param g
     * @param repo
     * @param cellPoint
     * @param width
     * @param height
     */
    void preDealBackground(Graphics g, Repository repo, Point cellPoint, int width, int height);

/**
 *
 *
 * @param g
 * @param shape
 */
void drawWithGradientLine(Graphics g, Shape shape);

/**
 *
 *
 * @param width
 * @param height
 */
void layoutDidChange(int width, int height);

/**
 *
 *
 * @param object
 * @return truefalse
 */
boolean equals(Object object);

/**
 *
 *
 * @param code
 * @return
 */
int fixHashCode(int code);

/**
 * JSON
 *
 * @return JSON
 * @throws JSONException
 */
JSONObject toJSONObject() throws JSONException;

/**
 *  h5
 * @param repo
 * @return
 * @throws JSONException
 */
JSONObject toJSONObject(Repository repo) throws JSONException;

JSONObject toJSONObject(Repository repo, Dimension size) throws JSONException;
```

```
    /**
     *
     *
     * @return
     * @throws CloneNotSupportedException
     */
    Object clone() throws CloneNotSupportedException;


    /**
     * web
     *
     * @return
     */
    String getBackgroundType();

    Background readAdditionalAttr(XMLableReader reader);

    void writeAdditionalAttr(XMLPrintWriter writer);
        /**
         *
         * @param operate excel,pdf,word...
         * @return
         */
        Background traverseForExport(ObjectHolder operate);

        /**
         * jsonweb.
         *
     * @param repo
         * @return json.
         */
    JSONObject createJSONConfig(Repository repo) throws JSONException;


    /**
     *
     * @param repo
     * @param width
     * @param height
     * @return
     * @throws JSONException
     */
    JSONObject createJSONConfig(Repository repo, int width, int height) throws JSONException;
}
```

ColorBackground.java

```
/*
 * Copyright(c) 2001-2010, FineReport Inc, All Rights Reserved.
 */
package com.fr.base.background;

import com.fr.general.Background;
import com.fr.general.ComparatorUtils;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.stable.StableUtils;
import com.fr.stable.web.Repository;
import com.fr.stable.xml.XMLPrintWriter;
import com.fr.stable.xml.XMLableReader;

import java.awt.*;
import java.awt.image.BufferedImage;

/**
 *
```

```
 */
public class ColorBackground extends AbstractBackground {
	private static final long serialVersionUID = -6930147321476711514L;

	private static java.util.Map initializeCBG = new java.util.HashMap();//<Color--ColorBackground>

	private Color color = null;

	/**
	 *
	 *
	 * @return
	 */
	public static ColorBackground getInstance() {
		return getInstance(null);
	}

	/**
	 *
	 *
	 * @param color
	 * @return
	 */
	public static ColorBackground getInstance(Color color) {
		// openjdk 1.8bug, hashmapnullkey,
		// https://bugs.openjdk.java.net/browse/JDK-8046085
		if (color == null) {
			return new ColorBackground(null);
		}

		Object valueBg = initializeCBG.get(color);
		if (valueBg != null) {
			return (ColorBackground) valueBg;
		} else {
			ColorBackground cbg = new ColorBackground(color);
			initializeCBG.put(color, cbg);
			return cbg;
		}
	}

	public ColorBackground() {

	}

	private ColorBackground(Color color) {
		this.color = color;
	}

	/**
	 *
	 *
	 * @return
	 */
	public Color getColor() {
		return color;
	}

	/**
	 *
	 *
	 * @param g
	 * @param shape
	 */
	public void paint(Graphics g, Shape shape) {
		Paint paint = this.getColor();
		if (paint == null) {
			return;
		}

		// shape Rectangle2D
		Graphics2D g2d = (Graphics2D) g;
```

```
        Paint oldPaint = g2d.getPaint();

        g2d.setPaint(paint);
        g2d.fill(shape);

        g2d.setPaint(oldPaint);
}

public JSONObject toJSONObject(Repository repo, Dimension size) throws JSONException {
        JSONObject jo = super.toJSONObject(repo, size);
        jo.put("color", StableUtils.javaColorToCSSColor(color));
        return jo;
}

protected BufferedImage createBufferedImage(int width, int height) {
        return null;
}

/**
 *
 *
 * @param g
 * @param shape
 */
public void drawWithGradientLine(Graphics g, Shape shape) {
        Paint paint = this.getColor();
        if (paint == null) {
                return;
        }

        // shape Rectangle2D
        Graphics2D g2d = (Graphics2D) g;
        Paint oldPaint = g2d.getPaint();

        g2d.setPaint(paint);
        g2d.draw(shape);

        g2d.setPaint(oldPaint);
}

/**
 *
 *
 * @param object
 * @return truefalse
 */
public boolean equals(Object object) {
        return object instanceof ColorBackground &&
                        ComparatorUtils.equals(((ColorBackground) object).color, color);
}

/**
 *hashcode,clone
 * stylemapget
 * @return hashcode
 */
public int hashCode() {
        return color == null ? 0 : color.hashCode();
}

/**
 *
 *
 * @param code
 * @return
 */
public int fixHashCode(int code) {
        return code ^ hashCode();
}

/**
```

```
         *
         *
         * @return
         * @throws CloneNotSupportedException
         */
        public Object clone() throws CloneNotSupportedException {
                ColorBackground cloned = (ColorBackground) super.clone();

                return cloned;
        }

        /**
         * JSON
         *
         * @return JSON
         * @throws JSONException
         */
        public JSONObject toJSONObject() throws JSONException {
                JSONObject js = super.toJSONObject();

                if (this.color != null) {
                        js.put("color", StableUtils.javaColor2JSColorWithAlpha(color));
                }

                return js;
        }

        /**
         * web
         *
         * @return
         */
        public String getBackgroundType() {
                return "ColorBackground";
        }

        /**
         *
         * @param reader
         * @return
         */
        public Background readAdditionalAttr(XMLableReader reader) {
                ColorBackground colorBackground = null; // kunsnat: . , Js, null
                if (reader.getAttrAsString("color", null) == null) {
                        colorBackground = ColorBackground.getInstance(null);
                } else {
                        colorBackground = ColorBackground.getInstance(reader.getAttrAsColor("color", Color.
black));
                }
                return colorBackground;
        }

        /**
         *
         * @param writer
         */
        public void writeAdditionalAttr(XMLPrintWriter writer) {
                writer.attr("name", "ColorBackground");
                if (getColor() != null) {
                        writer.attr("color", getColor().getRGB());
                }
        }

        @Override
        public JSONObject createJSONConfig(Repository repo) throws JSONException {
                Color color = this.getColor();
                if (color == null) {
                        // , {background : ""}.
                        return JSONObject.EMPTY;
                }
```

```
                String colorStr = StableUtils.javaColorToCSSColor(color);
                return JSONObject.create().put("background", colorStr);
        }

        public JSONObject createJSONConfig(Repository repo, int width, int height) throws JSONException{
                JSONObject jsonObject = new JSONObject();
                jsonObject.put("background-color", StableUtils.javaColorToCSSColor((this.getColor())));
                return jsonObject;
        }

}
```

ImageBackground.java

```
/*
 * Copyright(c) 2001-2010, FineReport  Inc, All Rights Reserved.
 */
package com.fr.base.background;

import com.fr.base.Base64;
import com.fr.base.BaseUtils;
import com.fr.base.GraphHelper;
import com.fr.base.ImageProvider;
import com.fr.general.Background;
import com.fr.general.ImageWithSuffix;
import com.fr.general.xml.FineImage;
import com.fr.general.xml.GeneralXMLTools;
import com.fr.json.JSONException;
import com.fr.json.JSONObject;
import com.fr.plugin.injectable.PluginModule;
import com.fr.stable.BaseConstants;
import com.fr.stable.Constants;
import com.fr.stable.StableUtils;
import com.fr.stable.StringUtils;
import com.fr.stable.bridge.StableFactory;
import com.fr.stable.fun.ImageLayoutDescriptionProcessor;
import com.fr.stable.plugin.ExtraClassManagerProvider;
import com.fr.stable.web.Repository;
import com.fr.stable.xml.XMLConstants;
import com.fr.stable.xml.XMLPrintWriter;
import com.fr.stable.xml.XMLReadable;
import com.fr.stable.xml.XMLableReader;

import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Image;
import java.awt.Paint;
import java.awt.Point;
import java.awt.Shape;
import java.awt.TexturePaint;
import java.awt.geom.Rectangle2D;
import java.awt.image.BufferedImage;
import java.awt.print.PrinterGraphics;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

import static com.fr.general.xml.GeneralXMLTools.DEFAULT_IMAGE_FORMAT;
import static java.awt.image.BufferedImage.TYPE_INT_ARGB;

/**
 *
 */
public class ImageBackground extends AbstractBackground {
    private transient ImageWithSuffix image = null;
    private int layout = Constants.IMAGE_CENTER; //layout.
    //layout,xml,,,layout
```

```
private int layout4Draw = Constants.IMAGE_CENTER;

private static final int PAINT_WIDTH_OFFSET = 40;
private static final int PAINT_HEIGHT_OFFSET = 40;

//
private Point cellPoint = new Point(-1, -1);

/**
 *
 */
public ImageBackground() {
    this(null);
}

/**
 *
 *
 * @param image
 * @deprecated
 */
public ImageBackground(Image image) {
    this(image, Constants.IMAGE_TILED);
}

/**
 *
 *
 * @param image
 * @param layout
 * @deprecated
 */
public ImageBackground(Image image, int layout) {
    this.setImage(ImageWithSuffix.build(image));
    this.setLayout(layout);
}

public ImageBackground(Image image, int layout, String path) {
    ImageWithSuffix imageWithSuffix = ImageWithSuffix.build(new FineImage(image, path));
    this.setImage(imageWithSuffix);
    this.setLayout(layout);
}

/**
 *
 *
 * @param image
 */
public ImageBackground(ImageWithSuffix image) {
    this(image, Constants.IMAGE_TILED);
}

/**
 *
 *
 * @param image
 * @param layout
 */
public ImageBackground(ImageWithSuffix image, int layout) {
    this.image = image;
    this.layout = layout;
}

/**
 *
 *
 * @return
 */
public Image getImage() {
    if (this.image == null) {
        return null;
```

```
    }

    return this.image.getImage();
}

/**
 * bufferImage
 *
 * @return
 */
private ImageProvider getFineImage() {
    if (this.image == null) {
        return null;
    }

    return this.image.getFineImage();
}

/**
 *
 *
 * @return
 */
public ImageWithSuffix getImageWithSuffix() {
    return this.image;
}

/**
 *
 *
 * @param img
 */
public void setImage(Image img) {
    this.image = ImageWithSuffix.build(img);
}

/**
 *
 *
 * @param image
 */
public void setImage(ImageWithSuffix image) {
    this.image = image;
}

/**
 *
 * Constants.IMAGE_DEFAULTConstants.IMAGE_TILED
 * Constants.IMAGE_CENTERConstants.IMAGE_EXTEND
 * Constants.IMAGE_DEFAULT
 *
 * @return
 */
public int getLayout() {
    return this.layout;
}

public int getLayout4Draw() {
    return layout4Draw == Constants.IMAGE_CENTER ? this.layout : layout4Draw;
}

public void setLayout4Draw(int layout4Draw) {
    this.layout4Draw = layout4Draw;
}

/**
 *
 *
 * @param layout
 * @see Constants#IMAGE_DEFAULT
 * @see Constants#IMAGE_TILED
```

```
     * @see Constants#IMAGE_CENTER
     * @see Constants#IMAGE_EXTEND
     */
    public void setLayout(int layout) {
        this.layout = layout;
    }

    public Point getCellPoint() {
        return cellPoint;
    }

    public void setCellPoint(Point cellPoint) {
        this.cellPoint = cellPoint;
    }

    /**
     *
     *
     * @param g
     * @param shape
     */
    @Override
    public void paint(Graphics g, Shape shape) {
        if (this.getImage() == null || shape == null) {
            return;
        }

        Graphics2D g2d = (Graphics2D) g;

        Paint paint = g2d.getPaint();

        if (isPDFExport(g2d) || isPrint(g2d)) {
            //pdfpdfpdf
            //pdf
            paintBackgroundImage(g2d, shape, StableUtils.isNotSupportARGB(g2d));
        } else {
            //
            g2d.setPaint(createPaint(shape, StableUtils.isNotSupportARGB(g2d)));
            g2d.fill(shape);
        }

        g2d.setPaint(paint);
    }

    private boolean isPDFExport(Graphics2D g2d) {
        return g2d.getClass().getName().contains("PdfGraphics2D");
    }

    private boolean isPrint(Graphics2D g2d) {
        return g2d instanceof PrinterGraphics;
    }

    /**
     *
     *
     * @param g
     * @param shape
     */
    @Override
    public void drawWithGradientLine(Graphics g, Shape shape) {
        if (this.getImage() == null || shape == null) {
            return;
        }
        Graphics2D g2d = (Graphics2D) g;

        Paint paint = g2d.getPaint();
        g2d.setPaint(createPaint(shape, StableUtils.isNotSupportARGB(g2d)));

        g2d.draw(shape);

        g2d.setPaint(paint);
```

```
    }

    /**
     *
     *
     * @param width
     * @param height
     */
    @Override
    public void layoutDidChange(int width, int height) {
        int imageLayout = layout;
        if (layout == 1) {
            imageLayout = StableUtils.changeImageLayout4Draw(getImage(), getLayout(), width, height);
        }
        setLayout4Draw(imageLayout);
    }

    private void paintBackgroundImage(Graphics2D g2d, Shape shape, boolean isNotSupportARGB) {
        Rectangle2D rec2D = shape.getBounds2D();
        if (rec2D.getWidth() <= 0 || rec2D.getHeight() <= 0) {
            return;
        }

        Image image = this.getImage();
        if (image instanceof BufferedImage) {
            dealWithNotDisplayProblemInARGB(g2d, (BufferedImage) image);
        }
        Shape oldClip = g2d.getClip();
        g2d.translate(rec2D.getX(), rec2D.getY());
        g2d.setClip(0, 0, (int) rec2D.getWidth(), (int) rec2D.getHeight());
        GraphHelper.paintImage(g2d, (int) rec2D.getWidth(), (int) rec2D.getHeight(), image,
                // .
                this.getLayout4Draw(), Constants.LEFT, Constants.TOP, -1, -1, isNotSupportARGB);
        g2d.translate(-rec2D.getX(), -rec2D.getY());
        g2d.setClip(oldClip);
    }

    private Paint createPaint(Shape shape, boolean isNotSupportARGB) {
        Rectangle2D rec2D = shape.getBounds2D();
        if ((int) rec2D.getWidth() <= 0) {
            rec2D.setRect(rec2D.getX(), rec2D.getY(), rec2D.getWidth() + PAINT_WIDTH_OFFSET, rec2D.getHeight());
        }
        if ((int) rec2D.getHeight() <= 0) {
            rec2D.setRect(rec2D.getX(), rec2D.getY(), rec2D.getWidth(), rec2D.getHeight() +
PAINT_HEIGHT_OFFSET);
        }
        //daniel:flashARGBBUGARGB653TexturePaintGraphHelper
        BufferedImage buffered = new BufferedImage((int) rec2D.getWidth(), (int) rec2D.getHeight(),
isNotSupportARGB ? BufferedImage.TYPE_INT_RGB : TYPE_INT_ARGB);
        Graphics2D g2 = buffered.createGraphics();
        GraphHelper.paintImage(g2, (int) rec2D.getWidth(), (int) rec2D.getHeight(), this.getImage(),
                // .
                this.getLayout4Draw(), Constants.LEFT, Constants.TOP, -1, -1, isNotSupportARGB);

        g2.dispose();

        dealWithNotDisplayProblemInARGB(g2, buffered);

        return new TexturePaint(buffered, rec2D);
    }

    private void dealWithNotDisplayProblemInARGB(Graphics2D g2d, BufferedImage buffered) {
        //pdf
        if (!g2d.getClass().getName().endsWith("PdfGraphics2D")) {
            return;
        }
        // "" ARGBPDF
        //
        int rgb = buffered.getRGB(0, 0);
        //-16777216
        // 0
```

```
    if (-16777216 == rgb || 0 == rgb) {
        buffered.setRGB(0, 0, rgb + 1);
    }
}

/**
 * Paints background of the shape and the background can move with the Scroll.
 * If the value is -1, use the actual size.
 * see paint(), this method has two more parameters.
 */
/**
 *
 *
 * @param g
 * @param shape
 * @param moveWidth
 * @param moveHeight
 */
public void paint4Scroll(Graphics g, Shape shape, int moveWidth, int moveHeight) {
    if (this.getImage() == null) {
        return;
    }

    Rectangle2D rec2D = shape.getBounds2D();

    Graphics2D g2d = (Graphics2D) g;
    Shape oldClip = g2d.getClip();
    g2d.clip(shape);
    g2d.translate(rec2D.getX(), rec2D.getY());

    GraphHelper.paintImageMoved(g2d, (int) rec2D.getWidth(), (int) rec2D.getHeight(), this.getImage(),
            //daniel
            this.getLayout(), Constants.LEFT, Constants.TOP, -1, -1,
            moveWidth, moveHeight, false);

    g2d.translate(-rec2D.getX(), -rec2D.getY());
    g2d.setClip(oldClip);
}

/**
 *
 *
 * @param obj
 * @return truefalse
 */
@Override
public boolean equals(Object obj) {
    if (!(obj instanceof ImageBackground)) {
        return false;
    }

    ImageBackground newImageBackground = (ImageBackground) obj;
    return this.getLayout() == newImageBackground.getLayout()
            && BaseUtils.imageEquals(this.getFineImage(), newImageBackground.getFineImage());
}

/**
 * JSON
 *
 * @return JSON
 * @throws JSONException
 */
@Override
public JSONObject toJSONObject() throws JSONException {
    JSONObject js = super.toJSONObject();

    js.put("layout", this.layout);
    //,
    if (this.getFineImage() != null) {
        js.put("image", Base64.encode(this.getFineImage().getBytes()));
        if (this.getImage() != null) {
```

```
            js.put("imgWidth", this.getImage().getWidth(null));
            js.put("imgHeight", this.getImage().getHeight(null));
        }
    }

    return js;
}

@Override
public JSONObject toJSONObject(Repository repo, Dimension size) throws JSONException {
    // MOBILE-10972 .
    JSONObject result = this.toJSONObject();
    result.put("src", createImageURL(repo));
    return result;
}

@Override
public JSONObject toJSONObject(Repository repo) throws JSONException {
    JSONObject result = this.toJSONObject();
    result.put("src", createImageURL(repo));
    return result;
}

/**
 * web
 *
 * @return
 */
@Override
public String getBackgroundType() {
    return "ImageBackground";
}

@Override
public Background readAdditionalAttr(XMLableReader reader) {
    //layout
    setLayout(reader.getAttrAsInt("layout", Constants.IMAGE_DEFAULT));

    reader.readXMLObject(new XMLReadable() {

        @Override
        public void readXML(XMLableReader reader) {
            if (reader.isChildNode()) {
                if (reader.getTagName().equals(XMLConstants.Image_TAG)) { //image.
                    Image image = GeneralXMLTools.readImage(reader);
                    setImage(ImageWithSuffix.build(image));
                } else if (XMLConstants.Deprecated_Image_TAG.equals(reader.getTagName())) {
                    Image image = GeneralXMLTools.deprecatedReadImage(reader);
                    setImage(ImageWithSuffix.build(image));
                } else if (ImageWithSuffix.XML_TAG.equals(reader.getTagName())) {
                    ImageWithSuffix image = new ImageWithSuffix();
                    setImage((ImageWithSuffix) reader.readXMLObject(image));
                }
            }
        }
    });
    return this;
}

@Override
public void writeAdditionalAttr(XMLPrintWriter writer) {
    writer.attr("name", "ImageBackground");

    //width/height
    writer.attr("layout", getLayout());

    //image.
    if (this.image != null) {
        this.image.writeXML(writer);
    }
}
```

```
    @Override
    public JSONObject createJSONConfig(Repository repo) throws JSONException {
        return createJSONConfig(repo, this.layout);
    }

    private JSONObject createJSONConfig(Repository repo, int layoutType) {
        StringBuffer sb = new StringBuffer();
        //
        String url = createImageURL(repo);
        sb.append("url(").append(url).append(") ");
        sb.append(createImageLayoutDescription(layoutType));
        if (layoutType == Constants.IMAGE_ADJUST || layoutType == Constants.IMAGE_EXTEND) {
            return adjustBackgroundJson(repo, url, sb, layoutType);
        }

        return JSONObject.create().put("background", sb.toString());
    }

    private String createImageURL(Repository repo) {
        //url
        BackgroundImageDisplayModeProcessor processor = StableFactory.getMarkedInstanceObjectFromClass
(BackgroundImageDisplayModeProcessor.XML_TAG,
                BackgroundImageDisplayModeProcessor.class);
        if (processor != null) {
            return processor.changeDisplayMode(repo, this.getFineImage(), BaseConstants.CHECKOUTIMAGE);
        }
        return repo.checkoutObject(this.getFineImage(), BaseConstants.CHECKOUTIMAGE);
    }


    private JSONObject adjustBackgroundJson(Repository repo, String url, StringBuffer sb, int type) throws
JSONException {
        JSONObject jo = JSONObject.create();
        // web
        if (repo.getBrowser().isLowIEVersion()) {
            jo.put("type", type);
            jo.put("url", url);
            // repoboxModel ie8
            jo.put("background", sb.toString());
            jo.put("background-repeat", "no-repeat");
            jo.put("background-image", "none");
            jo.put("filter", "progid:DXImageTransform.Microsoft.AlphaImageLoader(src='" + url + "',
sizingMethod='scale')");
            return jo;
        }

        jo.put("background", sb.toString());
        switch (type) {
            case Constants.IMAGE_ADJUST:
                jo.put("background-size", "contain");
                jo.put("background-position", "center");
                break;
            case Constants.IMAGE_EXTEND:
                jo.put("background-size", "100% 100%");
                break;
            default:
                break;
        }

        return jo;
    }

    private static String createImageLayoutDescription(int layout) {
        ExtraClassManagerProvider pluginProvider = PluginModule.getAgent(PluginModule.ExtraCore);
        if (pluginProvider != null) {
            ImageLayoutDescriptionProcessor processor = pluginProvider.getSingle
(ImageLayoutDescriptionProcessor.XML_TAG);
            if (processor != null) {
                String description = processor.getCustomLayoutDescription(layout);
                if (StringUtils.isNotEmpty(description)) {
```

```
                    return description;
                }
            }
        }

        switch (layout) {
            case Constants.IMAGE_DEFAULT:
                return "no-repeat";
            case Constants.IMAGE_ADJUST:
                return "no-repeat";
            case Constants.IMAGE_CENTER:
                return "no-repeat center";
            case Constants.IMAGE_TILED:
                return "repeat";
            case Constants.IMAGE_EXTEND:
                return "no-repeat center";
            default:
                return StringUtils.EMPTY;
        }
    }

    @Override
    public JSONObject createJSONConfig(Repository repo, int width, int height) throws JSONException {
        this.layoutDidChange(width, height);
        return this.createJSONConfig(repo, this.getLayout4Draw());
    }

    private void readObject(ObjectInputStream in) throws ClassNotFoundException, IOException {
        in.defaultReadObject();
        String format = in.readUTF();
        Object obj = in.readObject();
        if (obj instanceof ImageSerializable) {
            setImage(ImageWithSuffix.build(((ImageSerializable) obj).getImage(), format));
        }

    }

    private void writeObject(ObjectOutputStream out) throws IOException {
        out.defaultWriteObject();
        String format = this.image == null ? DEFAULT_IMAGE_FORMAT : this.image.getFormat();
        out.writeUTF(format);
        Image serialableTarget = getSeriableTarget(this.image);
        ImageSerializable imageSerializable = new ImageSerializable(serialableTarget, format);
        out.writeObject(imageSerializable);
    }

    private Image getSeriableTarget(ImageWithSuffix imageWithSuffix) {
        if (imageWithSuffix == null || imageWithSuffix.getImage() == null) {
            return new BufferedImage(1, 1, TYPE_INT_ARGB);
        }
        return imageWithSuffix.getImage();
    }
}
```

FR	8.0		
FR	9.0		
FR	10.0		
BI	3.6		
BI	4.0		
BI	5.1		

BI	5.1.2		
BI	5.1.3		

plugin.xml

```
<extra-designer>
        <BackgroundQuickUIProvider class="your class name"/>
</extra-designer>
```

BackgroundPane/BackgroundSpecialPaneFormBackgroundSettingPaneBackgroundQuickUIProvider

demodemo-background-quick-ui-provider