

SWTML Text Editor

This is an example of a full working TextEditor application using SWTML. The purpose of this example is to demonstrate how SWTML ties in with an application and how to separate your presentation. The example given is a fairly simple yet complete mini-framework for building SWTML driven applications quickly and easily. Once you understand the concepts of presentation separation, you should be able to apply them to any type of programming pattern. With that said, you may use the following code at your own discretion, and at your own risk.

Setup

Download the latest version of SWTML. The zip file contains all the necessary libraries to build SWTML application.

Download the Text Editor source-code [from here](#) .

texteditor.xml

```
<Panel border="1">  
    <Shell SWT.SHELL_TITLEBAR titled "<br></Panel>"<br></Shell>  
  
    <GridLayout<br></GridLayout>  
    <MenuBar<br></MenuBar>  
    <MenuItem<br></MenuItem>  
  
    <Menu<br></Menu>  
    <MenuItem<br></MenuItem>  
SWT.SelectionListener swtмл.examples.texteditor.FileMenuListener"  
  
</menu-item  
<MenuItem<br></MenuItem>  
SWT.SelectionListener swтмл.examples.texteditor.FileMenuListener"  
  
</menu-item  
<MenuItem<br></MenuItem>  
SWT.SelectionListener swтмл.examples.texteditor.FileMenuListener"  
  
</menu-item  
<MenuItem<br></MenuItem>  
SWT.SelectionListener swтмл.examples.texteditor.FileMenuListener"
```

```

</menu
</menu-item
<menu-item
    <menu-item "CADE"

<menu
    <menu-item "H"
    <menu-item "X"

SWT.SelectionListenerapps.swtмл.examples.texteditor.CutCopyPasteListener"
</menu-item
<menu-item "H"

    <menu-item "C"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.CutCopyPasteListener"
</menu-item
<menu-item "H"

    <menu-item "V"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.CutCopyPasteListener"
</menu-item
<menu-item "SEPARATOR"

<menu-item "H"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.FontListener"

</menu-item
<menu-item "CADE"
<menu

    <menu-item "H"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.ColorListener"

</menu-item
<menu-item "H"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.ColorListener"

</menu-item
<menu-item "H"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.ColorListener"

</menu-item
</menu
</menu-item
<menu-item "SEPARATOR"

<menu-item "H"
    SWT.SelectionListenerapps.swtмл.examples.texteditor.ClearListener"

</menu-item
</menu
</menu-item
</menu
<tool-bar

```

```

<tool-item
    id="bold"
    icon="/org/eclipse/swt/examples/texteditor/bold.jpg"
    style="fontWeight: bold"
    SWT.Selection org.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="italic"
    icon="/org/eclipse/swt/examples/texteditor/italic.jpg"
    style="fontStyle: italic"
    SWT.Selection org.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="underline"
    icon="/org/eclipse/swt/examples/texteditor/underline.jpg"
    style="text-decoration: underline"
    SWT.Selection org.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="strikeout"
    icon="/org/eclipse/swt/examples/texteditor/strikeout.jpg"
    style="text-decoration: line-through"
    SWT.Selection org.eclipse.swt.examples.texteditor.StyleListener"
</tool-bar
<styled-text
    style="SWT.BORDER | SWT.MULTI | SWT.V_SCROLL | SWT.H_SCROLL"
    id="styledText"
    background="white"
    foreground="black"
</styled-text

</shell

```

com.sheelapps.swtml.examples.texteditor.TextEditor.java :

```

package com.sheelapps;

import java.util.Vector;

import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.ExtendedModifyEvent;
import org.eclipse.swt.custom.ExtendedModifyListener;
import org.eclipse.swt.custom.StyleRange;
import org.eclipse.swt.custom.StyledText;

```

```
import org.eclipse.swt.graphics.Color;
import org.eclipse.swt.graphics.Point;

import org.eclipse.swt.graphics.RGB;
import org.eclipse.swt.graphics.Rectangle;
import org.eclipse.swt.widgets.Display;
import org.eclipse.swt.widgets.Shell;
import org.eclipse.swt.widgets.ToolItem;

import com.sheelapps.swtml.SWTObjectMap;
import com.sheelapps.swtml.SWTParser;

public class TextEditor {

    private static final String START_STYLES_MARK = "[[~Styles~]]"

    private static final Vector styledStyles = new Vector();

    private static final String NEWLINE = "\n";
    private static final String GET_PROPERTY = "GetProperty";

    private static Text styledText = null;

    private static Color RED = null;
    private static Color BLUE = null;
    private static Color GREEN = null;

    private static String fileName = null;

    private static Shell shell;

    private static String[] args;

    public static void main(String[] args) {
        TextEditor ed = new TextEditor();
        ed.open();
    }

    private void open() {
        Display display = Display.getDefault();

        try {
            initializeColors();
            SWTParser parser = new SWTParser();

            shell = new Shell(display, parser);
        } catch (Exception e) {
            e.printStackTrace();
        }

        shell.setText("Text Editor");
        shell.setSize(400, 300);
        shell.open();
        shell.setVisible(true);
    }

    private void initializeColors() {
        RED = new Color(display, new RGB(255, 0, 0));
        BLUE = new Color(display, new RGB(0, 0, 255));
        GREEN = new Color(display, new RGB(0, 255, 0));
    }

    private void createText() {
        styledText = new Text(shell, SWT.MULTI | SWT.WRAP | SWT.VERIFY_NONE);
        styledText.setText(START_STYLES_MARK);
    }

    private void createMenu() {
        MenuItem menu = new MenuItem(shell, SWT.NONE);
        menu.setText("File");
        menu.setMenu(new Menu(shell, SWT.NONE));
        menuItem = new MenuItem(menu, SWT.NONE);
        menuItem.setText("Open");
        menuItem.setListener(new Listener() {
            public void handleEvent(Event e) {
                open();
            }
        });
    }

    private void open() {
        fileName = JOptionPane.showInputDialog(shell, "Enter file name");
        if (fileName != null) {
            try {
                File file = new File(fileName);
                if (file.exists()) {
                    InputStream inputStream = new FileInputStream(file);
                    SWTParser parser = new SWTParser();
                    parser.parse(inputStream);
                    styledText.setText(parser.getStyledText());
                } else {
                    JOptionPane.showMessageDialog(shell, "File not found");
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    private void close() {
        shell.dispose();
    }
}
```

```

parentRect = display.getPrimaryMonitor
    .getBounds()
    .toRect()

size = shell.getSize
parentRect.x = parentRect.x + size.x / 2
parentRect.y = parentRect.y + size.y / 2

shell.open
shell.setSize
while !shell.isDisposed

if display.readAndDispatch
display.sleep
}
display.dispose

} catch {
// TODO Auto-generated catch block
e.printStackTrace

}

}

public StyledText getStyledText () {
    if styledText == null

        styledText = StyledText (SWTObjectMap.getAnObjectMap
            .get("styledText")
            .asExtendedModifyListenerStener)

    public void ExtendedModifyEvent e ) {
        handleExtendedModify e );
    }
};

}
return styledText ;

}

private void initializeColors () {
    Display display = Display.getDefault
    RED = new Color (display, new RGB (255, 0, 0));

    BLUE = new Color (display, new RGB (0, 0, 255))
    GREEN = new Color (display, new RGB (0, 255, 0))

}

private void handleExtendedModify ExtendedModifyEvent event ) {
    if event.length

```

```

return
StyleRange style ;
if event.length

|| styledText.event.start.equals
styledText.getLineDelimiter
// Have the new text take on the style of the text to its right

// (during
// typing) if no style information is active.
if caretOffset = styledText.getCaretOffset
style = null

if caretOffset < styledText.getCharCount
style = styledText.getCaretOffsetRangeAtOffset
if style = null

style = StyleRange )style.clone
style.start = event.start
style.length = event.length

}else
style = new StyleRange event.start, event.length,
SWT.NORMAL

}
if getButton("bold")Selection
style.f = SWT.BOLD

if getButton("italic")Selection
style.f = SWT.ITALIC
style.u = getButton("underline")Selection

style.s = getButton("getSelection")
if style.isUnstyled

styledText.setStyleRange
}else
// paste occurring, have text take on the styles it had when it was
// cut/copied

for( int i < cachedStyles.size ) {

style = StyleRange )cachedStyles.elementAt
StyleRange newStyle = StyleRange )style.clone

newStyle.start = style.start
newStyle.end = event.start
styledText.setStyleRange

}
}
}
}

```

```

sToolItem getItem (String
return (SWTObjectMap.get(getObjectMap
}

public void dispose () {
RED.dispose
GREEN.dispose

BLUE.dispose
shell.dispose

}
}

```

SWT Event Listeners :

ClearListener.java

```

package org.eclipse.swt.widgets;

import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.StyleRange;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.Event;
import org.eclipse.swt.widgets.Listener;

public class ClearListener implements Listener {

    public void handleEvent (Event e) {
        ITextEditor editor = (ITextEditor) e.getSource();
        ITextEditor sel = editor.getSelection();

        if (sel != null) {
            StyleRange style = new StyleRange(
                sel.getSelectionRange().x, sel.getSelectionRange().y, sel.getSelectionRange().length,
                SWT.NORMAL);

            editor.setStyleRange(style);
            editor.setSelectionRange(sel.getSelectionRange());
        }
    }
}

```

```
}
```

```
}
```

ColorListener.java

```
package editor
```

```
import org.eclipse.swt.SWT;  
import org.eclipse.swt.custom.StyleRange;  
import org.eclipse.swt.graphics.Color;  
import org.eclipse.swt.graphics.Point;  
import org.eclipse.swt.widgets.Event;  
import org.eclipse.swt.widgets.Listener;  
import org.eclipse.swt.widgets.MenuItem;
```

```
public class ColorListener implements
```

```
Listener {  
    private Event event;  
    private Color fg = null;
```

```
    private MenuItem item = null;  
    private boolean ignoreCase = true;
```

```
    public void handleEvent(Event event) {  
        this.event = event;  
        this.item = event.widget;  
        this.ignoreCase = true;
```

```
        if (event.detail == SWT.CUT) {  
            fg = TextEditor.RED;  
        }  
        else if (event.detail == SWT.PASTE) {  
            fg = TextEditor.GREEN;
```

```
        }  
        else if (event.detail == SWT.DROP) {  
            fg = TextEditor.BLUE;
```

```
        }  
        if (fg != null) {  
            StyleRange style = new StyleRange(0, 1, fg, null);  
            this.item.setText(style);
```

```
        }  
        return true;
```

```
    }  
    public void styleRange(StyleRange style, int range) {  
        if (style.fg != null) {  
            fg = style.fg;
```

```
        }  
        if (range == null) {  
            range = 0;
```



```

import org.eclipse.swt.SWT;
import org.eclipse.swt.custom.StyleRange;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.Event;
import org.eclipse.swt.widgets.Listener;
import org.eclipse.swt.widgets.Widget;

public class Listener implements Listener {

    public void handleEvent (Event event) {
        Widget widget = event.widget

        int sel = TextEditor.getSelectionRange();
        if (sel == null || sel.x == null || sel.y == null) {

            return;
        }
        StyleRange style = new StyleRange();
        style.start = sel.x;
        style.length = sel.y - sel.x;

        StyleRange range = TextEditor.getSelectionRangeAtOffset(sel.x);
        if (range == null) {

            style = new StyleRange(0, 1, SWT.NORMAL);

        }
        if (widget == TextEditor) {
            style.fontStyle = SWT.BOLD;
        }
        if (widget == TextEditor) {
            style.fontStyle = SWT.ITALIC;
        }
        if (widget == TextEditor) {
            style.underline = true;
        }
        if (widget == TextEditor) {
            style.strikeout = true;
        }
        TextEditor.setSelectionRange(sel.x, sel.y);
    }
}

```

SafeSaveDialog.java :

```
package org.eclipse.ui;

import java.io.File;
import org.eclipse.swt.SWT;
import org.eclipse.swt.widgets.FileDialog;
import org.eclipse.swt.widgets.MessageBox;
import org.eclipse.swt.widgets.Shell;

/**
 * This class provides a facade for the "save" FileDialog class. If the selected
 * file already exists, the user is asked to confirm before overwriting.
 */
public class SafeSaveDialog {

    // The wrapped FileDialog
    private FileDialog fd;
    private Shell parent;

    /**
     * SafeSaveDialog constructor
     *
     * @param shell the parent shell
     */
    public SafeSaveDialog(Shell shell) {
        fd = new FileDialog(shell, SWT.SAVE);
        parent = shell;

    }

    public boolean open() {
        // We store the selected file name in fileName
        String fileName = null;

        // The user has finished when one of the
        // following happens:
        // 1) The user dismisses the dialog by pressing Cancel
        // 2) The selected file name does not exist
        // 3) The user agrees to overwrite existing file
        boolean done = false;

        while (!done) {
            // Open the File Dialog
            fileName = fd.open();
            if (fileName == null) {
                // User has cancelled, so quit and return
                done = true;
            } else {

```

```

// User has selected a file; see if it already exists
if (correctExt fileName) {

done = false
continue
}
File = new File(fileName);

if (file.exists)
// The file already exists; asks for confirmation
MessageBox mb = new MessageBox dlg.get SWT.ALERT_WARNING

| SWT.YES | SWT.NO

// We really should read this string from a
// resource bundle
mb.fileName = "already exists. Do you want to replace it?"

// If they click Yes, we're done and we drop out. If
// they click No, we redisplay the File Dialog
done = mb.open == SWT.YES
} else

// File does not exist, so drop out
done = true
}
}
}
fileName;

}

protected (String fileName) {
return
}

private String getFileName () {
return getFileName
}

private List<String> getFileNames () {
return getFileNames
}

private List<String> getFilterExtensions () {
return getFilterExtensions
}

```

```

public FilterNames () {
    return FilterNames
}

public FilterPath () {
    return FilterPath
}

public FileName (string) {
    dlg.fileName = name
}

public FilterExtensions (string[]) {
    dlg.extensions = extensions
}

public FilterNames (string[]) {
    dlg.names = filterNames
}

public FilterPath (string) {
    dlg.filterPath = path
}

public ShellGetParent () {
    return parent
}

public Style () {
    return style
}

public GetText () {
    return text
}

public SetText (string) {
    dlg.setText = text
}
}
}

```

Rendered UI : [Text Editor](#)