

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_SHELL_TITLE titled "</Panel>  
  
<grid-layout  
<menu-item BAR "  
<menu-item CASCADE "  
  
<menu  
    <menu-item H "  
SWT.SelectionListeners.swtмл.examples.texteditor.FileMenuListener"  
</menu-item  
    <menu-item H "  
SWT.SelectionListeners.swtмл.examples.texteditor.FileMenuListener"  
</menu-item  
    <menu-item H "  
SWT.SelectionListeners.swtмл.examples.texteditor.FileMenuListener"  
</menu-item  
    <menu-item H "  
SWT.SelectionListeners.swtмл.examples.texteditor.FileMenuListener"  
</menu-item
```

```
</menu-item
</menu-item
<menuItem "CADE"

<menu
<menuItem "H"
= "Celerator+ X"

SWT.SelectionListenerapps.swt.html.examples.texteditor.CutCopyPasteListener"
</menu-item
<menuItem "H"
= "Celerator+ C"
SWT.SelectionListenerapps.swt.html.examples.texteditor.CutCopyPasteListener"
</menu-item
<menuItem "H"
= "Celerator+ V"
SWT.SelectionListenerapps.swt.html.examples.texteditor.CutCopyPasteListener"
</menu-item
<menuItem "SEPARATOR"

<menuItem "H"
SWT.SelectionListenerapps.swt.html.examples.texteditor.FontListener"
</menu-item
<menuItem "CADE"
<menu
<menuItem "H"
SWT.SelectionListenerapps.swt.html.examples.texteditor.ColorListener"
</menu-item
<menuItem "H"
SWT.SelectionListenerapps.swt.html.examples.texteditor.ColorListener"
</menu-item
<menuItem "H"
SWT.SelectionListenerapps.swt.html.examples.texteditor.ColorListener"
</menu-item
</menu
</menu-item
<menuItem "SEPARATOR"

<menuItem "H"
SWT.SelectionListenerapps.swt.html.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
    com.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="italic"
    icon="/org/eclipse/swt/examples/texteditor/italic.jpg"
    style="fontStyle: italic"
    SWT.Selection
    com.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="underline"
    icon="/org/eclipse/swt/examples/texteditor/underline.jpg"
    style="text-decoration: underline"
    SWT.Selection
    com.eclipse.swt.examples.texteditor.StyleListener"
<tool-item
    id="strikeout"
    icon="/org/eclipse/swt/examples/texteditor/strikeout.jpg"
    style="text-decoration: line-through"
    SWT.Selection
    com.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.swtмл.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 {

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

    static Vector<Color> cachedStyles = new Vector<Color>();

    public static String newline = System.getProperty("line.separator");

    static String styledText = null;

    static Color RED = null;

    static Color BLUE = null;

    static Color GREEN = null;

    static String fileName = null;

    static Shell shell;

    public static void main(String[] args) {

        TextEditor ed = new TextEditor ();
        ed.open();
    }

    public void open() {

        Display display = Display.getDefault();

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

            shell = new Shell (parser
                .getTextEditor().class,
                "%ResourceAppSource\\swtml\\examples\\texteditor\\texteditor.xml"
            );
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

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.get (this, ObjectMap
            .get (StyledText,
                StyledText, new ExtendedModifyListener ())

        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, 255,
        0));
    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.underline = getButton("underline")Selection

style.start = 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

}
}
}
}

```

```

public void getButton (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();
            style.x = sel.getSelectionRange().x;
            style.y = sel.getSelectionRange().y;
            style.length = sel.getSelectionRange().length;
            style.style = SWT.NORMAL;

            editor.setStyleRange(style);
        }
    }
}

```

```
}
```

```
}
```

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.CANCEL) {  
            return;  
        }  
        if (event.detail == SWT.CLICK) {  
            this.ignoreCase = true;
```

```
            if (event.detail == SWT.CLICK) {  
                this.ignoreCase = true;
```

```
                if (event.detail == SWT.CLICK) {  
                    this.ignoreCase = true;
```

```
                    if (event.detail == SWT.CLICK) {  
                        return;  
                    }  
                }  
            }  
        }  
    }  
}
```

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

```
        if (event.detail == SWT.CANCEL) {  
            return;  
        }  
        if (event.detail == SWT.CLICK) {  
            this.ignoreCase = true;
```



```
elif item.get("last") != ignoreCase
```

```

    TextEditor.getStyleText
  }
}

```

```
private CutCopy () {
```

```

    TextEditor. changedStyle
    sel = TextEditor. getStyledText . Range

```

```
int start = sel.x;
for (int i = sel.x; i < sel.y; i++) {
```

```
StyleRange style = TextEditor.getStyleRangeAtOffset(
    i, style == null
```

```
style.setStyle.startX;  
if (TextEditor.canUndo) {  
    UndoManager.undo();  
}
```

```
StyleRange lastStyle = StyleRange()
if (lastElement != null) {
    lastStyle = lastElement.getStyleRange()
}
```

```
&lastStyle.start + lastStyle.length - style.start  
lastStyle.length
```

```
}else
  TextEditor.setStyleElements
}
}
}else
```

```

    TextEditor.setStyle(styles)
  }
}
}

```

}

}

StyleListener.java

partager les données

```

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;

        StyleRange sel = TextEditor.getSelectedTextRange();
        if (sel == null || sel.x == 0) {

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

        StyleRange range = TextEditor.getSelectedTextRange();
        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.setStyleRange(style);
    }
    TextEditor.getSelectedTextRange();
}

```

SafeSaveDialog.java :

```
package editor
```

```
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.
 */
```

```
SafeSaveDialog {
```

```
// The wrapped FileDialog
```

```
protected
```

```
Shell parent;
```

```
/**
```

```
 * SafeSaveDialog constructor
```

```
 *
```

```
 * @param shell the parent shell
```

```
 */
```

```
SafeSaveDialog(Shell shell) {
```

```
    dlg = new FileDialog(shell, SWT.SAVE);
```

```
    parent = shell;
```

```
}
```

```
boolean open() {
```

```
    // We store the selected file name in fileName
```

```
    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
```

```
    done = false
```

```
    while (!done) {
```

```
        // Open the File Dialog
```

```
        fileName = dlg.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.ERROR_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.FilterNames = names
}

```

```

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

```

```

public GetParent () {
    return Parent
}

```

```

public Style () {
    return Style
}

```

```

public GetText () {
    return Text
}

```

```

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

```

Rendered UI : [Text Editor](#)