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

```
<?xml version="1.0" encoding="UTF-8"?>
 <shell style="SWT.SHELL_TRIM" text="TextEditor - Untitled">

 <grid-layout/>
 <menu style="SWT.BAR">
 <menu-item style="SWT.CASCADE" text="File">

<menu>
 <menu-item style="SWT.PUSH" text="New"
SWT.Selection="com.sheelapps.swtml.examples.texteditor.FileMenuListener">

</menu-item>
 <menu-item style="SWT.PUSH" text="Open"
SWT.Selection="com.sheelapps.swtml.examples.texteditor.FileMenuListener">

</menu-item>
 <menu-item style="SWT.PUSH" text="Save"
SWT.Selection="com.sheelapps.swtml.examples.texteditor.FileMenuListener">

</menu-item>
 <menu-item style="SWT.PUSH" text="Exit"
SWT.Selection="com.sheelapps.swtml.examples.texteditor.FileMenuListener"

 accelerator="SWT.ALT + x">
</menu-item>
```

```
</menu>
</menu-item>
 <menu-item style="SWT.CASCADE" text="&Edit">

<menu>
 <menu-item style="SWT.PUSH" text="Cu&t"
 accelerator="SWT.CTRL + X"

SWT.selection="com.sheelapps.swtml.examples.texteditor.CutCopyPasteListener"
</menu-item>
 <menu-item style="SWT.PUSH" text="&Copy"

 accelerator="SWT.CTRL + C"
SWT.selection="com.sheelapps.swtml.examples.texteditor.CutCopyPasteListener"
</menu-item>
 <menu-item style="SWT.PUSH" text="&Paste"

 accelerator="SWT.CTRL + V"
SWT.selection="com.sheelapps.swtml.examples.texteditor.CutCopyPasteListener"
</menu-item>
 <menu-item style="SWT.SEPARATOR"

 <menu-item style="SWT.PUSH" text="&Font"
SWT.selection="com.sheelapps.swtml.examples.texteditor.FontListener"

</menu-item>
 <menu-item style="SWT.CASCADE" text="&Color">
<menu>

 <menu-item style="SWT.PUSH" text="&Red"
SWT.selection="com.sheelapps.swtml.examples.texteditor.ColorListener"

</menu-item>
 <menu-item style="SWT.PUSH" text="&Green"
SWT.selection="com.sheelapps.swtml.examples.texteditor.ColorListener"

</menu-item>
 <menu-item style="SWT.PUSH" text="&Blue"
SWT.selection="com.sheelapps.swtml.examples.texteditor.ColorListener"

</menu-item>
</menu>
</menu-item>
 <menu-item style="SWT.SEPARATOR"

 <menu-item style="SWT.PUSH" text="C&lear"
SWT.selection="com.sheelapps.swtml.examples.texteditor.ClearListener"

</menu-item>
</menu>
</menu-item>
</menu>
<tool-bar
```

```
<tool-item
  image="/com/sheelapps/swtml/examples/texteditor/bold.jpg"
  style="SWT.CHECK" id="boldItem" toolTipText="Bold"
  SWT.Selection="com.sheelapps.swtml.examples.texteditor.StyleListener"/>
<tool-item
  image="/com/sheelapps/swtml/examples/texteditor/italic.jpg"
  style="SWT.CHECK" id="italicItem" toolTipText="Italic"
  SWT.Selection="com.sheelapps.swtml.examples.texteditor.StyleListener"/>
<tool-item
  image="/com/sheelapps/swtml/examples/texteditor/underline.jpg"
  style="SWT.CHECK" id="underlineItem" toolTipText="Underline"
  SWT.Selection="com.sheelapps.swtml.examples.texteditor.StyleListener"/>
<tool-item
  image="/com/sheelapps/swtml/examples/texteditor/strikeout.jpg"
  style="SWT.CHECK" id="strikeoutItem" toolTipText="Strikeout"
  SWT.Selection="com.sheelapps.swtml.examples.texteditor.StyleListener"/>
</tool-bar>
<styled-text
  style="SWT.BORDER | SWT.MULTI | SWT.V_SCROLL | SWT.H_SCROLL"
  id="styledText">
  <grid-data h-align="SWT.FILL" v-align="SWT.FILL" h-grab="true"
   v-grab="true"/>
</styled-text>
</shell>
```

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

```
package com.sheelapps.swtml.examples.texteditor;

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 cachedStyles = new Vector();


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

private static StyledText 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();
}

private void open(){

Display display = Display.getDefault();

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

shell = (Shell)parser
.doParse(TextEditor.class
.getResourceAsStream("/com/sheelapps/swtml/examples/texteditor/texteditor.xml"));
Rectangle parentRect = display.getPrimaryMonitor().getBounds();

Point size = shell.getSize();
shell.setLocation(parentRect.x + (parentRect.width - size.x) / 2,
parentRect.y + (parentRect.height - size.y) / 2);

shell.open();
shell.setSize(800, 600);
while (!shell.isDisposed()) {

if (!display.readAndDispatch())
display.sleep();
}
display.dispose();

} catch (Exception e) {
// TODO Auto-generated catch block
e.printStackTrace();

}

}

public static StyledText getStyledText() {
if (styledText == null) {

styledText = (StyledText) SWTObjectMap.getInstance().getObjectMap()
.get("styledText");
styledText.addExtendedModifyListener(new ExtendedModifyListener() {

public void modifyText(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 static void handleExtendedModify(ExtendedModifyEvent event) {

if (event.length == 0)
```

```
return;
StyleRange style;
if (event.length == 1

|| styledText.getTextRange(event.start, event.length).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.
int caretOffset = styledText.getCaretOffset();
style = null;

if (caretOffset < styledText.getCharCount())
style = styledText.getStyleRangeAtOffset(caretOffset);
if (style != null) {

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

} else {
style = new StyleRange(event.start, event.length, null, null,
SWT.NORMAL);

}
if (getButton("bold").getSelection())
style.fontStyle |= SWT.BOLD;

if (getButton("italic").getSelection())
style.fontStyle |= SWT.ITALIC;
style.underline = getButton("underline").getSelection();

style.strikeout = getButton("strikeout").getSelection();
if (!style.isUnstyled())

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

for (int i = 0; i < cachedStyles.size(); i++) {

style = (StyleRange) cachedStyles.elementAt(i);
StyleRange newStyle = (StyleRange) style.clone();

newStyle.start = style.start + event.start;
styledText.setStyleRange(newStyle);

}
}

}
```

```
static ToolItem getButton (String id ){

return (ToolItem )SWTObjectMap.getObjectMap().get(id);
}


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

BLUE.dispose();
shell.dispose();

}
}
```

SWT Event Listeners :

ClearListener.java

```
package com.swt.examples.texteditor;


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 event ){
Point sel = TextEditor.getStyledText().getSelectionRange();

if((sel != null) && (sel.y != 0)){

StyleRange style;
style = new StyleRange(sel.x, sel.y, null, null, SWT.NORMAL);

TextEditor.getStyledText().setStyleRange(style);
}
TextEditor.getStyledText().setSelectionRange(sel.x + sel.y, 0);
}

}
```

ColorListener.java

```
package org.eclipse.swt.examples.texteditor;

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 {

public void handleEvent(Event event){
Color fg=null;

MenuItem item = (MenuItem) event.widget;
if(item.getText().equalsIgnoreCase("Red")){

fg = TextEditor.RED;
}
else if(item.getText().equalsIgnoreCase("Green")){

fg = TextEditor.GREEN;
}
else if(item.getText().equalsIgnoreCase("Blue")){

fg = TextEditor.BLUE;
}

Point sel = TextEditor.getStyledText().getSelectionRange();

if((sel == null) || (sel.y == 0))
return;

StyleRange style, range;
for(int i = sel.x; i < sel.x + sel.y; i++){

range = TextEditor.getStyledText().getStyleRangeAtOffset(i);
if(range != null){
```

```
style = (StyleRange)range.clone();
style.start = i;
style.length = 1;

style.foreground = fg;
}else{
style = new StyleRange(i, 1, fg, null, SWT.NORMAL);

}
TextEditor.getStyledText().setStyleRange(style);
}
TextEditor.getStyledText().setSelectionRange(sel.x, sel.y);


}


}
```

CutCopyPasteListener.java

```
package texteditor;


import java.util.Vector;

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.MenuItem;

public class CutCopyPasteListener implements Listener{


public void handleEvent(Event event){
MenuItem item = (MenuItem)event.widget;

if(item.getText().equalsIgnoreCase("Cut")){
handleCutCopy();
TextEditor.getStyledText().cut();

}
else if(item.getText().equalsIgnoreCase("Copy")){
handleCutCopy();

TextEditor.getStyledText().copy();
}
else if(item.getText().equalsIgnoreCase("Paste")){
TextEditor.getStyledText().paste();
}
}

private void handleCutCopy(){

TextEditor.cachedStyles = new Vector();
Point sel = TextEditor.getStyledText().getSelectionRange();

int startX = sel.x;
for(int i = sel.x; i <= sel.x + sel.y - 1; i++){

StyleRange style = TextEditor.getStyledText().getStyleRangeAtOffset(i);
if(style != null){

style.start = style.start - startX;
if(!TextEditor.cachedStyles.isEmpty()){

StyleRange lastStyle = (StyleRange)TextEditor.cachedStyles
.lastElement();
if(lastStyle.similarTo(style)

&& lastStyle.start + lastStyle.length == style.start){
lastStyle.length++;

}else{
TextEditor.cachedStyles.addElement(style);
}
}else{

TextEditor.cachedStyles.addElement(style);
}
}
}

}


}
```

StyleListener.java

```
package com.[illegible]itor;
```

```java
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 StyleListener implements Listener{


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

Point sel = TextEditor.getText().getSelectionRange();
if((sel == null) || (sel.y == 0))

return;
StyleRange style;
for(int i = sel.x; i < sel.x + sel.y; i++){

StyleRange range = TextEditor.getText().getStyleRangeAtOffset(i);
if(range != null){

style = (StyleRange) range.clone();
style.start = i;
style.length = 1;

}else{
style = new StyleRange(i, 1, null, null, SWT.NORMAL);

}
if(widget == TextEditor.getButton("Bold")){
style.fontStyle ^= SWT.BOLD;

}else if(widget == TextEditor.getButton("Italic")){
style.fontStyle ^= SWT.ITALIC;

}else if(widget == TextEditor.getButton("Underline")){
style.underline = !style.underline;

}else if(widget == TextEditor.getButton("Strikeout")){
style.strikeout = !style.strikeout;

}
TextEditor.getText().setStyleRange(style);
}
TextEditor.getText().setSelectionRange(sel.x + sel.y);



}



}
```

SafeSaveDialog.java :

```java
package com.examples.texteditor;

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
protected FileDialog dlg;
protected Shell parent;
/**
   * SafeSaveDialog constructor
   *
   * @param shell the parent shell
   */

public SafeSaveDialog (Shell shell ){
dlg = new FileDialog(shell, SWT.SAVE);
parent = shell;

}

public String 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 = 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 file = new File(fileName);

if (file.exists()) {
// The file already exists; asks for confirmation
MessageBox mb = new MessageBox(dlg.getParent(), SWT.ICON_WARNING

| SWT.YES | SWT.NO);

// We really should read this string from a
// resource bundle
mb.setMessage(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;
}
}
}
return fileName;

}

protected boolean correctExt(String fileName) {
return true;

}

public String getFileName() {
return dlg.getFileName();
}



public String[] getFileNames() {
return dlg.getFileNames();
}



public String[] getFilterExtensions() {
return dlg.getFilterExtensions();
}
```

```
public String[] getFilterNames(){
return dlg.getFilterNames();
}


public String getFilterPath(){
return dlg.getFilterPath();
}


public void setFileName(String string){
dlg.setFileName(string);

}

public void setFilterExtensions(String[] extensions){
dlg.setFilterExtensions(extensions);

}

public void setFilterNames(String[] names){
dlg.setFilterNames(names);

}

public void setFilterPath(String string){
dlg.setFilterPath(string);

}

public Shell getParent(){
return dlg.getParent();
}


public int getStyle(){
return dlg.getStyle();
}


Âpublic String getText(){
return dlg.getText();
}


public void setText(String string){
dlg.setText(string);

}
}
```

Rendered UI : Text Editor