

JFormattedText Field with Date

Contributed by Aminur Rashid
 Wednesday, 25 March 2009
 Last Updated Tuesday, 07 April 2009

While trying JFormattedTextField, I had this requirement to show placeholder to the user to guide him providing the date input. There was also a need to pop up user with the error message, if the input is correct. (Yeah, I understand that many prefer inline error messages than the pop-ups but then requirements are requirements.)

One way could be to provide a MaskFormat to the JFormattedTextField and use directly validates the date, yourself using the pattern.

What I tried is used both the DateFormat and MastFormat with the JFormattedTextField. The code can be modified for individual needs but the idea is to provide an input verifier to the JFormattedTextField. Use the formatter set for JFormattedTextField to verify the inputs. Along with these, also install a mask formatter to show placeholders to user.

Code to create a JFormattedTextField to verify dateinputs:

```
package com.aminur.swing;

import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;

import java.text.DateFormat;
import java.text.ParseException;
import java.text.SimpleDateFormat;

import java.util.Date;

import javax.swing.InputVerifier;
import javax.swing.JComponent;
import javax.swing.JFormattedTextField;
import javax.swing.JOptionPane;
import javax.swing.text.DateFormatter;
import javax.swing.text.MaskFormatter;

/*
 * @author Aminur Rashid (aminur ^ aminurrashid.com)
 *
 * Usage
 * DateFormattedTextFieldVerifier.getFormattedTextField("dd/MM/yyyy",true,true,"####/####/####","_/_/____",new
Character('_'))
 * or in case you do not want to use mask
 * DateFormattedTextFieldVerifier.getFormattedTextField("dd/mm/yyyy", true,false,null,null,null)
 * DateFormattedTextFieldVerifier.getFormattedTextField("dd/mm/yyyy");
 */
public class DateFormattedTextFieldVerifier {

    private String mask = null;// e.g "####/####/####";
    private SimpleDateFormat format;
    private DateFormatter df;
    protected String datePattern;// = "dd/MM/yyyy";
    private String placeholder;// e.g "_/_/____";
    private char placeholderChar; //e.g '_'
    private DateVerifier objDateVerifier;
    private JFormattedTextField dateTextField;
    private String patternString;

    public static JFormattedTextField getFormattedTextField(String pattern) {
```

```

    return (DateFormattedTextFieldVerifier.getFormattedTextField(pattern, true,false,null,null,null));
}

/**
 * @param pattern
 * @param showPopUp
 * @return
 */
public static JFormattedTextField getFormattedTextField(String pattern, boolean showPopUp,boolean installMask,
String mask, String maskPlaceholder, Character placeholder) {
    DateFormattedTextFieldVerifier thisClass = new DateFormattedTextFieldVerifier(pattern, showPopUp,installMask,
mask, maskPlaceholder, placeholder);
    return thisClass.dateTextField;
}
/*****
 * Private methods
 *****/

private DateFormattedTextFieldVerifier(String pattern, boolean showErrorPopUp, boolean installMask, String mask,
String maskPlaceholder, Character placeholderChar) {
    patternString = pattern;
    format = new SimpleDateFormat(pattern);
    df = new DateFormatter(format) {
        public Object stringValueToValue(String string) throws ParseException {
            if (string == null || string.length() == 0 || string.equals(placeholder)) {
                return null;
            }
            return super.stringValueToValue(string);
        }
    };
    format.setLenient(true);
    dateTextField = new JFormattedTextField(df);
    if(installMask){
        placeholder = maskPlaceholder;
        this.mask = mask;
        this.placeholderChar = placeholderChar.charValue();
        installMask(dateTextField);
        dateTextField.setFocusLostBehavior(JFormattedTextField.PERSIST);
        dateTextField.addFocusListener(new FocusAdapter() {

            public void focusGained(FocusEvent e) {
                installMask(dateTextField);
            }
        });
    }
    objDateVerifier = new DateVerifier(format, placeholder, showErrorPopUp, installMask);
    dateTextField.setInputVerifier(objDateVerifier);
}

private void installMask(JFormattedTextField tfObject) {
    MaskFormatter objMask = new MaskFormatter() {
        public void install(JFormattedTextField ftf) {
            String currentText = ftf.getText();
            super.install(ftf);
            if (ftf != null) /*&& !ftf.isEditValid()*/ {
                if (currentText != null && currentText.length() > 0) {
                    ftf.setText(currentText);
                }
            }
        }
    }
}

public String valueToString(Object obj) throws ParseException {

```

```

        if (obj instanceof Date) {
            return (format.format((Date)obj));
        }
        return super.valueToString(obj);
    }
};
try {
    objMask.setMask(mask);
    objMask.setAllowsInvalid(true);
    objMask.setPlaceholderCharacter(placeholderChar);
    objMask.install(tfObject);
} catch (Exception pe) {
    System.out.println("Exception " + pe);
}
}

class DateVerifier extends InputVerifier {
    private DateFormat dateFormat = null;

    private boolean isPopupErrorEnable = true;
    private String formatPlaceholder = null;
    private boolean useMask;
    DateVerifier(DateFormat df, String fPlaceholder, boolean isPopupErrorEnable, boolean useMask) {
        this.dateFormat = df;
        this.isPopupErrorEnable = isPopupErrorEnable;
        this.useMask = useMask;
        formatPlaceholder = fPlaceholder;
    }

    private boolean isValid(String p_Date) throws ParseException {

        try {
            dateFormat.parse(p_Date);
            return true;
        } catch (ParseException dfe) {
            throw dfe;
        }
    }

    public boolean verify(JComponent input) {
        try {
            if (input instanceof JFormattedTextField) {
                JFormattedTextField jtf = (JFormattedTextField)input;
                String currentValue = jtf.getText();
                if (currentValue == null || currentValue.equalsIgnoreCase(formatPlaceholder) || currentValue.length() == 0) {
                    jtf.setValue(null);
                    jtf.commitEdit();
                    //uncomment if you want to see placeholder when not in focus as well
                    /*if(useMask)
                        installMask(jtf);*/
                    return true;
                }
            }
            try {
                isValid(jtf.getText());
                jtf.commitEdit();
                return true;
            } catch (Exception e) {
                if (isPopupErrorEnable) {
                    JOptionPane.showMessageDialog(jtf.getParent(),
                        e.getMessage() + "\n Please specify date as " + patternString,
                        "Date Error..", JOptionPane.ERROR_MESSAGE);
                }
            }
            jtf.selectAll();
        }
    }
}

```

```
        return false;
    }
}
} catch (Exception ex) {
    ex.printStackTrace();
    return false;
}
return true;
}
}
}
```