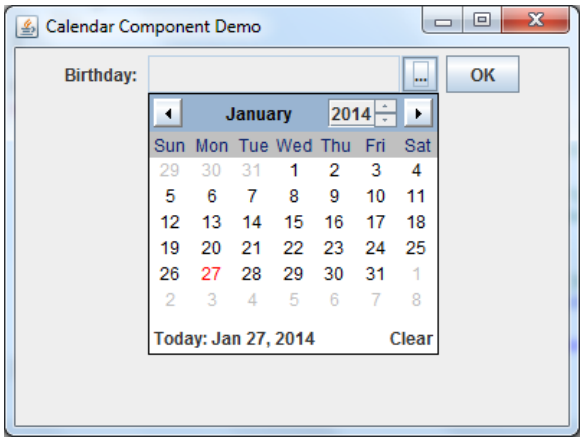
# How to use JDatePicker to display calendar component

Written by Nam Ha Minh
Last Updated on 05 July 2019 | Print Email

This tutorial shows you how to use the JDatePicker open-source library in order to display a calendar component in Java Swing programs with some necessary customizations. You will end up creating the following program:



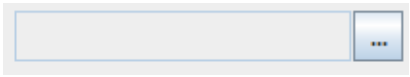

## 1. Quick start with JDatePicker

Click here to download the `JDatePicker` library from SourceForge. The latest version as of now is 1.3.2. Extract the downloaded archive `JDatePicker-1.3.2-dist.zip`, and then find and add the `jdatepicker-1.3.2.jar` file to your project's classpath.

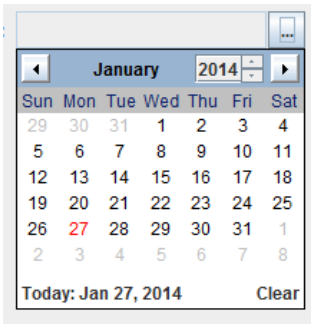
It's pretty simple to create and add a date picker component to a container, i.e. a JFrame. It involves in choosing an appropriate `DateModel` which is required by a `JDatePanelImpl` which is required by a `JDatePickerImpl` - which is then added to the container. For example, the following code snippet creates a date picker component using the `UtilDateModel`, and then adds it to the frame:

```
1  UtilDateModel model = new UtilDateModel();
2  JDatePanelImpl datePanel = new JDatePanelImpl(model);
3  JDatePickerImpl datePicker = new JDatePickerImpl(datePanel);
4
5  frame.add(datePicker);
```

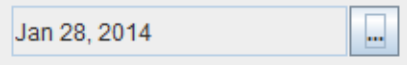
The result is a component displayed which looks like this:



It contains a disabled, read-only text field on the left, and an ellipsis button that will pop up a calendar when clicked:



Upon choosing a date, the calendar dismisses and the selected date is filled into the text field:




## 2. Dealing with Date Models

The `JDatePicker` library provides three date models which correspond to three date time types in Java:

- `UtilDateModel`: the date picker will return the selected date as an object of type `java.util.Date`.
- `CalendarDateModel`: the date picker will return the selected date as an object of type `java.util.Calendar`.
- `SqlDateModel`: the date picker will return the selected date as an object of type `java.sql.Date`.

Choosing which model is depending on your need. And notice that, depending on the model used,

the `JDatePickerImpl` returns the selected date object of appropriate type. For example, the following statement gets the selected date in case the model is `UtilDateModel`:

```
1  Date selectedDate = (Date) datePicker.getModel().getValue();
```

For the `CalendarDateModel` model, use the following code:

```
1  Calendar selectedValue = (Calendar) datePicker.getModel().getValue();
2  Date selectedDate = selectedValue.getTime();
```

For the `SqlDateModel` model, use the following code:

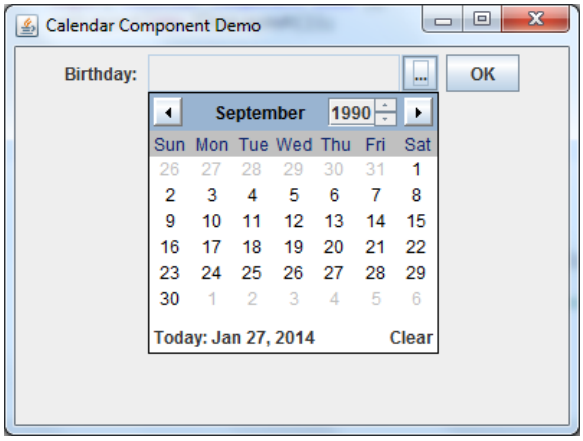
```
1  java.sql.Date selectedDate = (java.sql.Date) datePicker.getModel().getValue();
```

## 3. Setting initial date

You can set the initial date for the calendar component when it is popped up. For example:

```
1  UtilDateModel model = new UtilDateModel();
2  model.setDate(1990, 8, 24);
```

That sets the initial date to September 24, 1990 (because in Java, the month number is zero-based). Result:




If you want to set initial date for the text field, use the following statement:

```
model.setSelected(true);
```

Result:


# 4. Customizing the date format

The default format of the date shown in the text field may not suite your need. In such case, you can create your own class that extends the `javax.swing.JFormattedTextField.AbstractFormatter` class. For example:

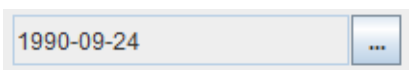
```
package net.codejava.swing;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import javax.swing.JFormattedTextField.AbstractFormatter;

public class DateLabelFormatter extends AbstractFormatter {

    private String datePattern = "yyyy-MM-dd";
    private SimpleDateFormat dateFormatter = new SimpleDateFormat(datePattern);

    @Override
    public Object stringToValue(String text) throws ParseException {
        return dateFormatter.parseObject(text);
    }

    @Override
    public String valueToString(Object value) throws ParseException {
        if (value != null) {
            Calendar cal = (Calendar) value;
            return dateFormatter.format(cal.getTime());
        }

        return "";
    }

}
```

As you can see, this class overrides the `stringToValue()` method to parse a `String` to a `Date` object; and overrides the `valueToString()` method to format the `Calendar` object to a `String`. The date pattern to use is **yyy-MM-dd**.

And pass an instance of this custom class when constructing the date picker component as follows:

```
JDatePickerImpl datePicker = new JDatePickerImpl(datePanel, new DateLabelFormatter());
```

Result:




You can download the Java source files under the attachment section.

## References:

- JDatePicker Home Page (SourceForge)