NachoCalendar manual

This is a tiny manual. If you have more questions please read the javadoc.

1- Quickstart: How implement it in your application.

At first you must add nachocalendar-xx.jar to your classpath. Factory classes are provided for easy startup.

Using DateField:

First add the imports (if you like):

```
import net.sf.nachocalendar.CalendarFactory;
import net.sf.nachocalendar.components.DateField;
```

Then you need a new instance, so you can use static method from the factory "createDateField()":

```
.....
.....
DateField datefield = CalendarFactory.createDateField();
.....
```

Finally add it to your form, as any other component. It inherits from Jpanel:

```
.....
form.add(datefield);
........
```

Using DatePanel

First add the imports (if you like):

```
import net.sf.nachocalendar.CalendarFactory;
import net.sf.nachocalendar.components.DatePanel;
```

Then create the instance and add it to your form:

```
DatePanel datepanel = CalendarFactory.createDatePanel();
form.add(datepanel);
```

There are 2 constructors, the default and other which lets you specify if you want to show the week numbers, in this case the code would be:

```
DatePanel datepanel = CalendarFactory.createDatePanel(true);
form.add(datepanel);
```

Using CalendarPanel

First (again) add the imports:

```
import net.sf.nachocalendar.CalendarFactory;
import net.sf.nachocalendar.components.DatePanel;
```

Then create the instance and add it to your form:

```
CalendarPanel calendarpanel = CalendarFactory.createCalendarPanel();
form.add(calendarpanel);
```

```
This will create a 3 months calendar. Again there are others
constructors, you can specify how many months you want, for example:
```

```
CalendarPanel calendarpanel = CalendarFactory.createCalendarPanel(5)
```

This will create a 5 months CalendarPanel

Setting and getting the date:

The the three components follow the JformattedTextField style. You have to use the get/set Value with an Object as parameter. This Object must be instanceof java.util.Date (so a java.util.Date will be returned)

```
// setting the date
datefield.setValue(new Date());
// getting the selected date
Date choosed = (Date) datefield.getValue();
```

Multiple selection:

CalendarPanel and DatePanel support multiple selection, enabled by default. This is made using a DateSelectionModel. Three selection modes are supported:

```
DateSelectionModel.SINGLE_SELECTION: only one date at time can be
selected.

DateSelectionModel.SINGLE_INTERVAL: only a continuous date interval
can be selected.

DateSelectionModel.MULTIPLE_INTERVAL: multiple date intervals or
single dates can be selected (Default)
```

Example:

```
import net.sf.nachocalendar.CalendarFactory;
import net.sf.nachocalendar.model.DateSelectionModel;
import net.sf.nachocalendar.components.CalendarPanel;
////////////
CalendarPanel cp = CalendarFactory.createCalendarPanel();
cp.setSelectionMode(DateSelectionModel.SINGLE_SELECTION;
```

//////////

To obtain the dates selected, both components have the method "Object[] getValues()", it returns an Object array.

That's all.

2-Properties:

Common Properties:

The components have some properties you can change, some can be selected only in the instantiation. Unless specified, they can be changed anytime.

ShowWeekNumber:

Selfexplanatory. Only can be specified in the constructor.

HeaderRenderer:

Specifies the renderer used for the header. More in next chapters

Model

Specifies the data model used to connect dates with objects. More next...

Renderer:

Specifies the renderer used for the days. More in next chapters

FirstDayOfWeek:

Specifies the first day of the week (sunday or monday). Defaults to the machine locale

WorkingDays:

It sets the working days. Non working days are renderer with other background

AntiAliased:

It sets the antialiased property.

PrintMoon

Sets the moon phase capability

ShowToday

Shows/hides the today button

DateField only properties:

ShowOkCancel:

Applies only to DateField. If true, when the monthpanel is showing a pair of ok/cancel buttons are provided to accept/cancel the selection. If false, the selection is made with a single mouse click.

AllowsInvalid:

Applies only to DateField. If true, you can type invalid dates, 35 days months for example and when de datefield losts its focus, this is corrected. If false it doesn't permits invalid dates, but is more difficult to use. Can be changed after construction.

CalendarPanel only properties:

Quantity

Quantity of months to show at once

Orientation:

Posible values: CalendarPanel.VERTICAL (default) or CalendarPanel.HORIZONTAL

ScrollPosition:

It can be HORIZONTAL/LEFT or VERTICAL/RIGHT , it depends on the orientation.

YearPosition

It can be HORIZONTAL/UP or VERTICAL/DOWN, it depends again on the orientation.

EternalScroll

Sets the eternal scroll property.

3-JTable support:

A DateField can be used inside a JTable for date selection. On this purpose a TableCellEditor and TableCellRenderer are provided. There are 2 ways to use them: as default editor/renderer or as editor/renderer for a TableColumn. To make it easy, there is a utility class (JTableCustomizer). It has convenient static methods.

First method:

```
import net.sf.nachocalendar.table.JTableCustomizer;

/////////

JTable table = new JTable();
JTableCustomizer.setDefaultEditor(table);
```

This will show a DateField when a cell has a java.util.Date object.

Second method:

```
import net.sf.nachocalendar.table.JTableCustomizer;


/////////


JTable table = new JTable();
JTableCustomizer.setEditorForRow(table, 1);
```

This will always use a DateField for all the cells in the second column.

4-Customizers:

From version 0.23 there is a new way to configure the components. There are “customizers”, used to set the properties of a components by reading a text file (.properties or .xml). Right now, if you place a file called “nachocalendar.properties” or “nachocalendar.xml” in the classpath, the components created by the CalendarFactory will be customized acording to the settings defined in the file. The format of the .properties file is the usual:

```
firstDayOfWeek=SUNDAY
allowsInvalid=false
antiAliased=false
```

The format of the xml file is the following:

```
<nachocalendar>
          <property name="firstDayOfWeek">SUNDAY</property>
          <property name="allowsInvalid">false</property>
          <property name="antiAliased">false</property>
....
</nachocalendar>
```

See the examples in the resources.

Also, you can have many customizers in your application, using the new classes found in the package net.sf.nachocalendar.customizer.

Example:

```
CustomizerFactory factory = new CustomizerFactory(“myfile.properties”);
DateField datefield = factory.createDateField();
```

5-Customizations:

More to come...