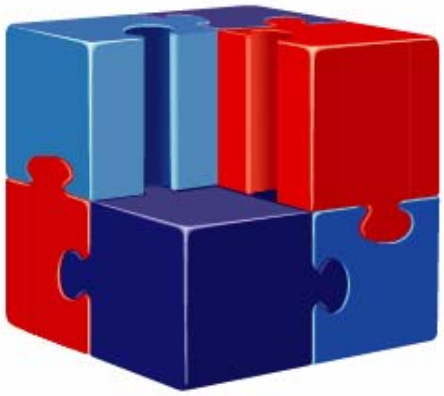
jProductivity LLC

Protect your investments with Components!



Components!

Calendar

User Guide

http://www.jproductivity.com

Contents

Highlights

Components! Calendar is a Java Component which can be integrated in a Java IDE and used in a GUI Java Application or Applet. Main purpose of the Calendar component is to present user with the single or multiple months view. Calendar is presented in a user-friendly way as a table of the calendar days.

Components! Calendar is a robust and sophisticated component for JFC/Swing:

- Essential for creating modern powerful and polished Java GUI application.
- Designed for speed and flexibility with advanced features and powerful performance
- Provide comprehensive presentation layer allowing developer to focus on the implementation
- Designed with Developer Needs in mind.
- Over the last several years millions of developers have embraced Java as their primary or secondary language. Wealth of available Java IDEs, utilization of Java in the development of Server-based, Desktop, Web and Mobile applications is a clear indication that Java is hear to stay. Java's strong support for the component-based development created a growing market of developers that come to rely on the component-based development approach. Utilization of JavaBeans components introduced rapid development of feature reach GUI interfaces and development cost efficiency.
- Built-in convenience.
- All jProductivity components design with the concept allowing developers to simply drop components on the form and start using it. However if customization is required, all jProductivity components are highly customizable and extendible.
- Customized display options.
- All aspects of a visual component could be highly customized. From simple Background and Foreground Color manipulation to complete replacement of the Components internal rendering engine with the custom one.
- On the fly validation.
- No mater what data type component is designed to work with, all jProductivity components validate its data on the fly and therefore protecting developer and end-user from accidental data corruption.
- Wide range of applications.
- Because all jProductivity Components are true 100% Java they could be used in any visual application environment – from web to desktop application.

High Level Description

The following topics briefly outline major Components! Calendar features and abilities.

Basic Features

Main purpose of the Calendar component is to present user with the single or multiple months view. Calendar is presented in a user-friendly way as a table of the calendar days.

In the dates table view each Calendar's column has an associated caption displaying the weekday. Each Calendar's row shows one whole week of the weekdays. Previous and

next month's days are also could be shown in the Calendar. Presentation of the off-month days and its navigation are optional and can be disabled either at design or at run time.

Navigation within the Calendar accomplished by any of the following methods:

- Keyboard's arrows, [Shift+] Tab to navigate be one day
- Home/End buttons to select First/Last day of current month
- Mouse click to select any visible day
- Mouse wheel to navigate by month. In combination with Ctrl button it will navigate by year, with Shift button – by week
- Page Up/Page Down buttons to navigate by month. In combination with Ctrl button it will navigate by year, with Shift button – by week
- Using special Calendar's navigation buttons allowing navigation by week, month and year (depending on the Calendar's type)

When off-month day is selected the Calendar is switched to the next/previous month (depending on the selected day) and selected day is highlighted.
The Calendar is locale-dependent. Some tool tips, text, days and weekday names and their order depend on the selected locale. For example for US locale Sunday will be first day of the week, for Russian local it will be last.

There are some general Calendar's properties which can be changed from their default values and are essential for Calendar:

- Locale – allows displaying Calendar in the way appropriate to current or selected locale, like weekday name and their order, or date format for the Date Field.
- Short Day Caption – shows only first letter from the week day's name
- Show Grid – shows grid dividing Calendar's days
- Show Vertical Lines – shows vertical lines dividing days of the week
- Show Horizontal Lines - shows horizontal lines dividing weeks
- Show Off-month Days – shows off-month days
- Allow Selection Off-month Days – allows to select off-month days
- Preserve Current Day – if Allow Selection Off-month Days is disabled and off-month day is selected by mouse or keyboard, new selection will be refused and restored to the previous one if preserve, otherwise the nearest day of current month is selected (in case of next month - last day of the month, in case of previous month - first day of the month)
- Foreground – foreground color (only for current month weekday days)
- Background – background color
- Header Foreground – week day names foreground color
- Weekend Foreground – weekend foreground color
- Weekend Background – weekend background color
- Off-month Weekday Foreground – off-month weekday foreground color
- Off-month Weekday Background – off-month weekday background color
- Off-month Weekend Foreground - off-month weekend foreground color
- Off-month Weekend Background - off-month weekend background color
- Today Border Color – today's border color
- Today Border Thickness - today's border thickness

Each Calendar's day can be represented in a different way depending on the user needs by manipulating required day's properties such as Font Style (Bold or/and Italic, Plain), Border, Foreground, Background, Icon, Tool Tip Text , etc..

There are 3 types of the Calendar and the Date Field as described below.

Simple Month Calendar

Simple Month Calendar represents a month calendar with navigation by keyboard and mouse only.

Month Calendar with Navigation

Month Calendar with Navigation extends Simple Month Calendar adding two decoration panels on the top and bottom of the Calendar. By default there are Navigation Panel on the top of the Calendar and Control Panel on the bottom. Both of them can be replaced be custom panel or deleted completely.
Navigation Panel shows selected month and year in the format of the given locale. There are six navigation buttons assembled into two groups which allow navigating through the Calendar. User presented with ability to navigate by week, by month or by year. Each navigation button can be hidden.
Control Panel contains two control buttons. First button shows today's date in the format of given locale. By clicking on this button the Calendar will be set to today's date (based on the system date). Second button, titled with "OK", generates an event showing that the date is selected. Each control button can be hidden.

Multiple Month Calendar

Multiple Month Calendar is similar to Month Calendar with Navigation, but allows showing as many months as required. The amount of month and rows can be specified. It could be very useful if required to present user with a quarter or a whole year. Months in the Calendar can be arranged in a vertical/horizontal line or in the grid by manipulating two properties - Calendars Amount and Rows. First month specifies initial starting Month of the Multi Month Calendar Component. For example, if Calendar is required to show whole year, January should specified as a first month.

Date Field with Popup Month Calendar

Date Field with Popup Month Calendar represents a field to show, enter and validate the date. The Date Field consists of the Text Field to show/enter the date and the Button to popup the Month Calendar with Navigation. The Month Calendar with Navigation can be presented by pressing Down Arrow key on the keyboard (can be redefined). Using popup calendar allows selecting a particular date rather then typing it. The text, tool tip and icon for the Button can be changed as well.
Date format depends on given locale. Year digits amount can be changed to show two or four digits. Entered date will be validated against specified locale. Date Field Component validates that month should be between 1 and 12 and date between 1 and 28(31) (depending on the month and year). If the date part is incorrect or incomplete it will be corrected to the nearest min/max value. For example, incorrect value February 29, 2003 (2/29/03) will be corrected to February 28, 2003 (2/28/03).

Samples and How-To…

This section will answer on the most frequently asked question "*How to...?"* and provides the samples of Calendar usage.

How to change days rendering in the calendar?

There are two ways this could be accomplished:

Implementing CalendarDayRendererListener

Simplest way is to implement CalendarDayRendererListener and listen to `onCalendarDayRenderer` event. This event is triggered for each visible day in the Calendar. The Calendar passes tree parameters to this event: the Source Object (an instance of `CalendarInterface`), the Date and current day's renderer Component. Current implementation of the `CalendarDayRenderer` passes `JLabel` component. Any desired property of this component can be changed to show the day of the month in a desired way.

Bellow is a sample illustrating use of the `CalendarDayRendererListener`:

```
public class MyClass
    implements CalendarDayRendererListener
{
  public void onCalendarDayRenderer(CalendarInterface aSource, Date aDate,
                                    Component aRendererComponent)
  {
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(aDate);

    // If first day of month change aRendererComponent's tool tip text and
    // background color
    if (calendar.get(Calendar.DAY_OF_MONTH) == 1)
    {
      ((JLabel) aRendererComponent).setToolTipText("First Day of the Month");
      aRendererComponent.setBackground(Color.re);
    }
  }
}
```

Overriding CalendarDayRenderer

Second approach is to replace `DefaultCalendarDayRenderer` for `CalendarPanel` and/or `MultipleMonthCalendarDayRenderer` for `MultipleMonthCalendarPanel`. A new renderer must implement `CalendarDayRenderer` interface and assign it (set) to the Calendar by its `setCalendarDayRenderer()` method. The new renderer can extend any type of Java Component. If a renderer is designed for `MultipleMonthCalendarDayRenderer` it must implement `MultipleMonthCalendarDayRenderer` interface.

Bellow is a sample of the `CalendarDayRenderer` which extends `JButton` which used to present a Calendar's day:

```
public class ButtonCalendarDayRenderer
    extends JButton
    implements CalendarDayRenderer
{
  protected static Font TODAY_FONT = null;

  public ButtonCalendarDayRenderer()
  {
    super();

    setOpaque(true);
    setMargin(new Insets(0, 0, 0, 0));
```

```
  }

  public Component getRendererComponent(CalendarPanel aCalendarPanel,
                                        CalendarItem aCalendarItem,
                                        boolean anIsSelected,
                                        boolean aHasFocus)
  {
    setFont(aCalendarPanel.getTable().getFont());
    setForeground(aCalendarPanel.getForeground());
    setBackground(aCalendarPanel.getBackground());
    setText(aCalendarItem.toString());

    if (aCalendarItem.isWeekend())
    {
      if (aCalendarItem.isCurrentMonth())
      {
        setForeground(aCalendarPanel.getWeekendForeground());
        setBackground(aCalendarPanel.getWeekendBackground());
      }
      else
      {
        setForeground(aCalendarPanel.getOffMonthWeekendForeground());
        setBackground(aCalendarPanel.getOffMonthWeekendBackground());
      }
    }
    else
    {
      if (!aCalendarItem.isCurrentMonth())
      {
        setForeground(aCalendarPanel.getOffMonthWeekdayForeground());
        setBackground(aCalendarPanel.getOffMonthWeekdayBackground());
      }
    }
    if (anIsSelected)
    {
      setBackground(aCalendarPanel.getTable().getSelectionBackground());
    }
    if (aCalendarItem.isToday())
    {
      if (TODAY_FONT == null)
        TODAY_FONT = new Font(aCalendarPanel.getTable().getFont().getName(),
                              Font.BOLD,
                              aCalendarPanel.getTable().getFont().getSize());
      setFont(TODAY_FONT);
    }

    return this;
  }
}

public class MyClass
{
  CalenarPanel calenarPanel = new CalenarPanel();

  public MyClass()
  {
    calenarPanel.setCalendarDayRenderer(new ButtonCalendarDayRenderer());
  }
}
```

How to delete/replace decoration panels in the MonthCalendarPanel?

If developer would like to replace existing decoration panels with his/her own, developer should extend abstract class `CalendarDecorationPanel`; add desired components and business logic; place it on the panel with `MonthCalendarPanel` and assign (set) it to the `MonthCalendarPanel's` desired location (top or bottom) using corresponding methods `setBottomDecoration()` or `setTopDecoration()`. Abstract class `CalendarDecorationPanel` has the field `fCalendarPanel`, which can be used to get access to its `MonthCalendarPanel`. If developer needs to delete one of the decoration panels he should assign (set) `null` to a proper method.
Bellow is a sample illustrating how to delete a decoration panel:

```
monthCalendarPanelInstance.setBottomDecoration(null);
```

Bellow is a sample how to create one's own decoration penal and assign it to the `MonthCalendarPanel`:

```
public class MyDecorationPanel
    extends CalendarDecorationPanel
{
  //your code
}
monthCalendarPanelInstance.setBottomDecoration(new MyDecorationPanel());
```

How to arrange 4 calendars in a vertical or horizontal line in the MultipleMonthCalendarPanel?

To have a horizontal line of 4 calendars just set 4 to `setCalendarsAmount()` and 1 to `setRows()`. To have a vertical line of 4 calendars just set 4 to `setCalendarsAmount()` and 4 to `setRows()`.

```
multipleMonthCalendarPanelInstance.setCalendarsAmount(4);
multipleMonthCalendarPanelInstance.setRows(1);
```

How to arrange 4 calendars in the grid of size 2x2 in the MultipleMonthCalendarPanel?

Set 4 to `setCalendarsAmount()` and 2 to `setRows()`.

```
multipleMonthCalendarPanelInstance.setCalendarsAmount(4);
multipleMonthCalendarPanelInstance.setRows(2);
```

How to show whole year in the MultipleMonthCalendarPanel?

To present a whole year (12 month), set 12 to `setCalendarsAmount()`, 3 or 4 to `setRows()` to have the grid 4x3 or 3x4 respectively and 0 to `setFirstMonth()`. Setting 0 to `setFirstMonth()` will start the Calendar from January.

```
multipleMonthCalendarPanelInstance.setCalendarsAmount(12);
multipleMonthCalendarPanelInstance.setRows(4);
multipleMonthCalendarPanelInstance.setFirstMonth (0);   // January
```

How to control walking through the Calendar?

There are two ways this could be accomplished:

Having Calendar Controller of your Own

First approach is to create your own Calendar Controller and set it to the Calendar by its method `setCalendarController()`. New Calendar Controller should implement `CalendarController` interface. However, a better approach would be to extend abstract class `CalendarControllerAdapter` which already implements `CalendarController` and performs some basic navigation logic such as calculation of the next day and month and navigation within the current month.

Bellow is a sample how to create your own Calendar Controller extending `CalendarControllerAdapter`:

```
public class MyCalendarController
    extends CalendarControllerAdapter
{
  // invoked when navigated to previous month
  protected void previousMonthSelection(CalendarPanel aCalendarPanel, int
                                        aSelection)
  {
    // your ligic
  }
  // invoked when navigated to next month
  protected void nextMonthSelection(CalendarPanel aCalendarPanel, int aSelection)
  {
    // your logic
  }
}
```

Implementing CalendarDateListener Interface

This approach is better and much simpler to implement. Developer needs to implement `CalendarDateListener` interface and listen to `aboutToChangeDate` event. This event is triggered each time when the date is changed. The Calendar passes tree parameters to this event: the Source Object (an instance of `CalendarInterface`), old `Date` and new `Date`. You can use both dates for validation and calculate new date which must be set to the Source object. If new date was set to the source then this method must return true, otherwise false. This event can be used, for example, to disable navigation on Sunday. Next sample illustrates how to disable navigation on Sunday and, depending on the navigation direction, jump to the next date after Sunday or stay on the last selected date:

```
public class MyClass
    implements CalendarDateListener
{
  public boolean aboutToChangeDate(CalendarInterface aSource, Date aOldDate, Date
                                   aNewDate)
  {
    Calendar newDate = Calendar.getInstance(aSource.getLocale());
    newDate.setTime(aNewDate);
    Calendar newDateCopy = (Calendar) newDate.clone();
    boolean result = false;

    // If Sunday, calculate prev/next date
    if (newDate.get(Calendar.DAY_OF_WEEK) == Calendar.SUNDAY)
    {
      Calendar oldDate = Calendar.getInstance(aSource.getLocale());
      oldDate.setTime(aOldDate);
```

```
        // new date after old one, next direction, add one day, otherwise prev
        // direction, subtract one day
        if (newDate.after(oldDate))
        {
          newDate.add(Calendar.DAY_OF_MONTH, 1);

          // if selection of off-month days is disabled and month is changed
          // return selection to the last selected date
          if (!aSource.isAllowSelectOffMonthDays() &&
              newDateCopy.get(Calendar.MONTH) != newDate.get(Calendar.MONTH))
            newDate.add(Calendar.DAY_OF_MONTH, -2);
        }
        else
        {
          newDate.add(Calendar.DAY_OF_MONTH, -1);

          // if selection of off-month days is disabled and month is changed
          // return selection to the last selected date
          if (!aSource.isAllowSelectOffMonthDays() &&
              newDateCopy.get(Calendar.MONTH) != newDate.get(Calendar.MONTH))
            newDate.add(Calendar.DAY_OF_MONTH, 2);
        }

        aSource.setDate(newDate.getTime());

        result = true;
      }

      return result;
    }
  }
```

Feedback

As part of the continuing effort to improve our product, we welcome your comments, suggestions and general feedback regarding the product.

If you have questions about Components!, please feel free to contact us for further information at components@jproductivity.com or visit our site using the following URL: http://www.jproductivity.com.

If you discover any issues or defects in Components!, please send the description of them to components@jproductivity.com.