MiG Calendar JavaBeans

API

Ellen Bergdahl
MiG InfoCom AB 2007

Package
com.miginfocom.beans

com.miginfocom.beans
Class AbstractHeaderBean

```
java.lang.Object
   |-com.miginfocom.beans.AbstractBean
      |-com.miginfocom.beans.AbstractHeaderBean
```

All Implemented Interfaces:
Serializable

Direct Known Subclasses:
DateHeaderBean, CategoryHeaderBean

---

public abstract class **AbstractHeaderBean**
extends AbstractBean

Some basic functionality for header JavaBeans in the MiG Calendar component.

Method Summary

java.awt.Paint	getBackgroundPaint() Property: The background paint in the date area.
DateAreaBean	getContainer() Returns the container that this header is decorating.
int	getEdge() Property: The edge (top, left, bottom, right) that this header should be placed at.
int	getExpandToCorner() Property: What corner, if any, the header should expand into if (if it exist, or rather has any space).
boolean	isVisible() Property: If this header should be visible or not.
void	revalidateRepaintContainer() If there is a connected container, revaidate and repaint it
void	setBackgroundPaint(java.awt.Paint p) Property: The background paint in the date area.
boolean	setDateAreaContainer(DateAreaBean container) Sets the com.miginfocom.calendar.datearea.DateAreaContainer that this header should decorate.
void	setEdge(int edge) Property: The edge (top, left, bottom, right) that this header should be placed at.
void	setExpandToCorner(int corner) Property: What corner, if any, the header should expand into if (if it exist, or rather has any space).
void	setVisible(boolean b) Property: If this header should be visible or not.

Methods inherited from class com.miginfocom.beans.AbstractBean
addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods

getContainer

`public DateAreaBean getContainer()`

Returns the container that this header is decorating.

Returns:
The container that this header is decorating. Can be `null`.

revalidateRepaintContainer

`public void revalidateRepaintContainer()`

If there is a connected container, revaidate and repaint it

setDateAreaContainer

`public boolean setDateAreaContainer(DateAreaBean container)`

Sets the `com.miginfocom.calendar.datearea.DateAreaContainer` that this header should decorate. This will be called by the controller that handles them both and the header will not work unless this property has been set.

Parameters:
`container` - The container that this header should decorate. `null` resets the header.

Returns:
If the container was changed. `false` if the same container was set again.

getExpandToCorner

`public int getExpandToCorner()`

Property: What corner, if any, the header should expand into if (if it exist, or rather has any space). Values are:

1. `CORNER_EXPAND_NONE` - The header will not expand into any corner.
2. `CORNER_EXPAND_BOTH` - The header will expand into both adjacent corners.
3. `CORNER_EXPAND_BOTTOM_OR_RIGHT` - The header will expand into the bottom corner if this is a vertical (standing) header and to the right corner if it's a horizontal header.
4. `CORNER_EXPAND_TOP_OR_LEFT` - The header will expand into the top corner if this is a vertical (standing) header and to the left corner if it's a horizontal header.

Returns:
The current value. `DateAreaContainer.CORNER_EXPAND_NONE` is default.

setExpandToCorner

```
public void setExpandToCorner(int corner)
```

Property: What corner, if any, the header should expand into if (if it exist, or rather has any space). Values are:

1. `CORNER_EXPAND_NONE` - The header will not expand into any corner.
2. `CORNER_EXPAND_BOTH` - The header will expand into both adjacent corners.
3. `CORNER_EXPAND_BOTTOM_OR_RIGHT` - The header will expand into the bottom corner if this is a vertical (standing) header and to the right corner if it's a horizontal header.
4. `CORNER_EXPAND_TOP_OR_LEFT` - The header will expand into the top corner if this is a vertical (standing) header and to the left corner if it's a horizontal header.

This method will recreate the header if the value changes.

Parameters:

`corner` - The new value.

getEdge

```
public int getEdge()
```

Property: The edge (top, left, bottom, right) that this header should be placed at.

1. `SwingConstants.TOP` - The header will be above the date area.
2. `SwingConstants.LEFT` - The header will be to the left of the date area.
3. `SwingConstants.BOTTOM` - The header will be to the right of the date area.
4. `SwingConstants.RIGHT` - The header will be below the date area.

Returns:

The current value. `SwingConstants.TOP` is default.

setEdge

```
public void setEdge(int edge)
```

Property: The edge (top, left, bottom, right) that this header should be placed at.

1. `SwingConstants.TOP` - The header will be above the date area.
2. `SwingConstants.LEFT` - The header will be to the left of the date area.
3. `SwingConstants.BOTTOM` - The header will be to the right of the date area.
4. `SwingConstants.RIGHT` - The header will be below the date area.

This method will recreate the header if the value changes.

Parameters:

`edge` - The new edge.

getBackgroundPaint

```
public java.awt.Paint getBackgroundPaint()
```

(continued on next page)

(continued from last page)

Property: The background paint in the date area. Will override the normal background `Color` so that `Paint` objects can be used instead.

Note! This might be shown as the grid color. It really isn't but if the grid color is set to `null` and the cells have a background paint this background color will show through as the grid color. This is the way if some cells should span more than one cell.

Returns:

The current background paint. May be `null`.

---

setBackgroundPaint

```
public void setBackgroundPaint(java.awt.Paint p)
```

Property: The background paint in the date area. Will override the normal background `Color` so that `Paint` objects can be used instead.

Note! This might be shown as the grid color. It really isn't but if the grid color is set to `null` and the cells have a background paint this background color will show through as the grid color. This is the way if some cells should span more than one cell.

Parameters:

`p` - The new background paint. May be `null`.

---

isVisible

```
public boolean isVisible()
```

Property: If this header should be visible or not.

Returns:

If the header is corrently visible.

---

setVisible

```
public void setVisible(boolean b)
```

Property: If this header should be visible or not.

Parameters:

`b` - If the header should be visible.

com.miginfocom.beans
Class ActivityAShapeBean

```
java.lang.Object
   |-com.miginfocom.beans.AbstractBean
      |-com.miginfocom.beans.ActivityAShapeBean
```

All Implemented Interfaces:
Serializable

---

public class **ActivityAShapeBean**
extends AbstractBean

Encapsulates the default `com.miginfocom.ashape.shapes.RootAShape` that is a round rectangle with a title and a summary.

Uses the `RootAShape` gotten from calling `createDefault(int)` and configuring the properties of it.

Constructor Summary

public	ActivityAShapeBean()

Method Summary

void	addMouseInteractionListener(MouseInteractionListener l) Adds a listener that listens to all mouse events on this shape and all sub shapes that has a hit area report set to true.
void	addMouseInteractionListener(MouseInteractionListener l, boolean asWeakRef) Adds a listener that listens to all mouse events on this shape and all sub shapes that has a hit area report set to true.
void	addSubShape(AShape shape) Adds a generic sub shape to the root shape.
int	getAntiAlias() Property: The anti aliasing hint used when drawing the graphics for the shape.
java.awt.Paint	getBackground() Property: The background paint for the shape.
double	getCornerRadius() Property: The corner radius for the outline com.miginfocom.util.gfx.RoundRectangle.
boolean	getDraggable() Property: If the activity should AShape should be draggable and thus send out DefaultDateArea.AE_DRAG_PRESSED when pressed.
java.awt.Cursor	getMouseOverCursor() Property: The Cursor showed when the mouse is over the shape or null if none.

int	getMouseOverSummaryUnderline() Property: How thick the underline should be for mouse overs of on the summary text. 0 will disable it.
java.awt.Paint	getOutlinePaint() Property: The Paint used to draw the outline.
float	getOutlineStrokeWidth() Property: The width of the outline around the shape.
String	getPaintContext() Property: The context that the AShape that this bean represent will adhere to.
PlaceRect	getPlaceRect() Property: How this shape will relate to the reference rectangle (bounds) given by the activity layout system.
int	getPrimaryDimension() **Deprecated.** *Shunted to* getResizeHandles()
int	getResizeHandles() Property: The size the resize boxes should be placed. -1 if no resize boxes and thus there will be no way to resize the shape.
float	getShadowBlurRadius() Property: The blur radius for the blur filter.
double	getShadowCornerRadius() Property: The corner radius for the shadow com.miginfocom.util.gfx.RoundRectangle.
java.awt.Paint	getShadowPaint() Property: The Paint used to draw the shadow.
boolean[]	getShadowPaintOptimization() Property: The shadow is divided into a 3x3 grid.
PlaceRect	getShadowPlaceRect() Property: How the shadow will be placed relative to the main shape.
int	getShadowSliceSize() Property: How big (thick) the outer (border) slices for the shadow will be.
AtRefRangeNumber	getTextAlignX() Property: Where within the place rect for the text to draw the text string.
AtRefRangeNumber	getTextAlignY() Property: Where within the place rect for the text to draw the text string.
int	getTextAntiAlias() Property: The anti aliasing hint used when drawing the text for the shape.
java.awt.Font	getTextFont() Property: The font used to draw the main text in the shape.
java.awt.Paint	getTextForeground() Property: The Paint used to draw the main text.

PlaceRect	getTextPlaceRect() Property: How this shape's main text will relate to the reference rectangle (bounds) given by the activity layout system.
String	getTextTemplate() Property: The template text for the main text.
AtRefRangeNumber	getTitleAlignX() Property: Where within the place rect for the title to draw the title text.
AtRefRangeNumber	getTitleAlignY() Property: Where within the place rect for the title to draw the title text.
java.awt.Font	getTitleFont() Property: The font used to draw the title in the shape.
java.awt.Paint	getTitleForeground() Property: The Paint used to draw the title.
PlaceRect	getTitlePlaceRect() Property: How this shape's title will relate to the reference rectangle (bounds) given by the activity layout system.
String	getTitleTemplate() Property: The template text that will show in the title of the shape.
void	removeMouseInteractionListener(MouseInteractionListener l) Removes the listener if it exists locally.
void	removeSubShape(AShape shape) Removes the sub shape.
void	setAntiAlias(int hint) Property: The anti aliasing hint used when drawing the graphics for the shape.
void	setBackground(java.awt.Paint paint) Property: The background paint for the shape.
void	setCornerRadius(double radius) Property: The corner radius for the outline com.miginfocom.util.gfx.RoundRectangle.
void	setDraggable(boolean b) Property: If the activity should AShape should be draggable and thus send out DefaultDateArea.AE_DRAG_PRESSED when pressed.
void	setMouseOverCursor(java.awt.Cursor c) Property: The Cursor showed when the mouse is over the shape or null if none.
void	setMouseOverSummaryUnderline(int width) Property: How thick the underline should be for mouse overs of on the summary text. 0 will disable it.
void	setOutlinePaint(java.awt.Paint paint) Property: The Paint used to draw the outline.
void	setOutlineStrokeWidth(float width) Property: The width of the outline around the shape.

void	setPaintContext(String paintContext) Property: The context that the AShape that this bean represent will adhere to.
void	setPlaceRect(PlaceRect placeRect) Property: How this shape will relate to the reference rectangle (bounds) given by the activity layout system.
void	setPrimaryDimension(int dim) **Deprecated.** *Shunted to* (int)
void	setResizeHandles(int dim) Property: The size the resize boxes should be placed. -1 if no resize boxes and thus there will be no way to resize the shape.
void	setShadowBlurRadius(float r) Property: The blur radius for the blur filter.
void	setShadowCornerRadius(double radius) Property: The corner radius for the shadow com.miginfocom.util.gfx.RoundRectangle.
void	setShadowPaint(java.awt.Paint p) Property: The Paint used to draw the shadow.
void	setShadowPaintOptimization(boolean[] opt) Property: The shadow is divided into a 3x3 grid.
void	setShadowPlaceRect(PlaceRect pr) Property: How the shadow will be placed relative to the main shape.
void	setShadowSliceSize(int s) Property: How big the outer slices for the shadow will be.
void	setTextAlignX(AtRefRangeNumber x) Property: Where within the place rect for the text to draw the text string.
void	setTextAlignY(AtRefRangeNumber y) Property: Where within the place rect for the text to draw the text string.
void	setTextAntiAlias(int hint) Property: The anti aliasing hint used when drawing the text for the shape.
void	setTextFont(java.awt.Font font) Property: The font used to draw the main text in the shape.
void	setTextForeground(java.awt.Paint paint) Property: The Paint used to draw the main text.
void	setTextPlaceRect(PlaceRect placeRect) Property: How this shape's main text will relate to the reference rectangle (bounds) given by the activity layout system.
void	setTextTemplate(String s) Property: The template text for the main text.
void	setTitleAlignX(AtRefRangeNumber x) Property: Where within the place rect for the title to draw the title text.
void	setTitleAlignY(AtRefRangeNumber y) Property: Where within the place rect for the title to draw the title text.

void	setTitleFont(java.awt.Font font) Property: The font used to draw the title in the shape.
void	setTitleForeground(java.awt.Paint paint) Property: The Paint used to draw the title.
void	setTitlePlaceRect(PlaceRect placeRect) Property: How this shape's title will relate to the reference rectangle (bounds) given by the activity layout system.
void	setTitleTemplate(String s) Property: The template text that will show in the title of the shape.

Methods inherited from class com.miginfocom.beans.AbstractBean

addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

ActivityAShapeBean

public **ActivityAShapeBean**()

Methods

addSubShape

public void **addSubShape**(AShape shape)

Adds a generic sub shape to the root shape. It can for instance be used to add icons or other markers in some way that can not be expressed with the normalt properties. It is like adding a sub component to a Swing container.

Parameters:
shape - The shape to be added.

removeSubShape

public void **removeSubShape**(AShape shape)

Removes the sub shape.

Parameters:
shape - The shape to be removed.

getResizeHandles

public int **getResizeHandles**()

(continued from last page)

Property: The size the resize boxes should be placed. -1 if no resize boxes and thus there will be no way to resize the shape.

Can be `SwingConstants.VERTICAL` or `SwingConstants.HORIZONTAL` or -1.

Returns:

The current value. Default is `SwingConstants.VERTICAL`.

---

setResizeHandles

`public void setResizeHandles(int dim)`

Property: The size the resize boxes should be placed. -1 if no resize boxes and thus there will be no way to resize the shape.

Parameters:

`dim` - The new value. One of `SwingConstants.VERTICAL` or `SwingConstants.HORIZONTAL` or -1.

---

getPrimaryDimension

`public int getPrimaryDimension()`

Deprecated. *Shunted to* `getResizeHandles()`

Shunted to `getResizeHandles()`

---

setPrimaryDimension

`public void setPrimaryDimension(int dim)`

Deprecated. *Shunted to* `(int)`

Shunted to `setResizeHandles(int)`

---

getDraggable

`public boolean getDraggable()`

Property: If the activity should AShape should be draggable and thus send out DefaultDateArea.AE_DRAG_PRESSED when pressed.

Returns:

The current state.

---

setDraggable

`public void setDraggable(boolean b)`

Property: If the activity should AShape should be draggable and thus send out DefaultDateArea.AE_DRAG_PRESSED when pressed.

Parameters:

`b` - `true` means draggable. If same as set nothing will be done.

---

getMouseOverSummaryUnderline

`public int getMouseOverSummaryUnderline()`

Property: How thick the underline should be for mouse overs of on the summary text. 0 will disable it.

Returns:

The current width.

(continued from last page)

See Also:
`setMouseOverSummaryUnderline(int)`

setMouseOverSummaryUnderline

`public void setMouseOverSummaryUnderline(int width)`

Property: How thick the underline should be for mouse overs of on the summary text. 0 will disable it.

Parameters:
`width` - The width in pixels.

See Also:
`setMouseOverSummaryUnderline(int)`

getPaintContext

`public String getPaintContext()`

Property: The context that the AShape that this bean represent will adhere to. `com.miginfocom.calendar.activity.Activity` objects may have a `paintContext` set and contexts must match if the shape should be used to paint the activity.

For example there might be two types of activities, one with "bigShape" and one with "smallShape" as context. It is then possible to create two beans of this class, configure them differently, and set their respective paint context to either. This will paint some activities with one bean's and the others with the other bean's AShape.

Returns:
The current context. Default to `null`.

setPaintContext

`public void setPaintContext(String paintContext)`

Property: The context that the AShape that this bean represent will adhere to. `com.miginfocom.calendar.activity.Activity` objects may have a `paintContext` set and contexts must match if the shape should be used to paint the activity.

For example there might be two types of activities, one with "bigShape" and one with "smallShape" as context. It is then possible to create two beans of this class, configure them differently, and set their respective paint context to either. This will paint some activities with one bean's and the others with the other bean's AShape.

Parameters:
`paintContext` - The new context. May be `null`.

getMouseOverCursor

`public java.awt.Cursor getMouseOverCursor()`

Property: The Cursor showed when the mouse is over the shape or `null` if none.

Returns:
The current `Cursor`. May be `null`.

Since:
6.0

(continued from last page)

setMouseOverCursor

`public void setMouseOverCursor(java.awt.Cursor c)`

Property: The Cursor showed when the mouse is over the shape or `null` if none.

Parameters:

`c` - The new `Cursor`. May be `null` in which case the Cursor is not changed.

Since:

6.0

---

getShadowPaintOptimization

`public boolean[] getShadowPaintOptimization()`

Property: The shadow is divided into a 3x3 grid. Normally only the lower and right slices need to be drawn since the other slices will be fully covered (obscured) by the main shape. This greatly improves the performance. This must be turned of for certain shapes (and shadow offsets) and for instance if the main shape is translucent.

There are a few static arrays that can be used:

- `OPT_ALL` - Paints all slices.
- `OPT_BORDER` - Paints all but the middle slice.
- `OPT_BOTTOM_LEFT` - Paint bottom and left (five) slices.
- `OPT_BOTTOM_RIGHT` - Paint bottom and right (five) slices.
- `OPT_TOP_LEFT` - Paint top and left (five) slices.
- `OPT_TOP_RIGHT` - Paint top and right (five) slices.

Returns:

A nine element array with which slices are to be painted. The first element is upper left, the second upper middle and so on. `null` will paint all slices.

See Also:

`getShadowSliceSize()`

---

setShadowPaintOptimization

`public void setShadowPaintOptimization(boolean[] opt)`

Property: The shadow is divided into a 3x3 grid. Normally only the lower and right slices need to be drawn since the other slices will be fully covered (obscured) by the main shape. This greatly improves the performance. This must be turned of for certain shapes (and shadow offsets) and for instance if the main shape is translucent.

There are a few static arrays that can be used:

- `OPT_ALL` - Paints all slices.
- `OPT_BORDER` - Paints all but the middle slice.
- `OPT_BOTTOM_LEFT` - Paint bottom and left (five) slices.
- `OPT_BOTTOM_RIGHT` - Paint bottom and right (five) slices.
- `OPT_TOP_LEFT` - Paint top and left (five) slices.
- `OPT_TOP_RIGHT` - Paint top and right (five) slices.

Parameters:

`opt` - A nine element array with which slices are to be painted. The first element is upper left, the second upper middle and so on. `null` will paint all slices.

See Also:

(continued from last page)

`getShadowSliceSize()`

getShadowSliceSize

`public int getShadowSliceSize()`

Property: How big (thick) the outer (border) slices for the shadow will be. This is like the line thickness for LineBorder. The center slice (there 3x3) will be the remaining size.

Returns:
The current size.

See Also:
`setShadowPaintOptimization(boolean[])`

setShadowSliceSize

`public void setShadowSliceSize(int s)`

Property: How big the outer slices for the shadow will be. This is like the line thickness for LineBorder. The center slice (there 3x3) will be the remaining size.

Parameters:
`s` - The new size.

See Also:
`setShadowPaintOptimization(boolean[])`

getShadowPaint

`public java.awt.Paint getShadowPaint()`

Property: The `Paint` used to draw the shadow.

Returns:
The current `Paint`. May be `null`.

setShadowPaint

`public void setShadowPaint(java.awt.Paint p)`

Property: The `Paint` used to draw the shadow.

Parameters:
`p` - The new `Paint`. May be `null` in which case no shadow will be painted.

getShadowBlurRadius

`public float getShadowBlurRadius()`

Property: The blur radius for the blur filter. Greater means more smeared shadow but it takes longer to compute.

Returns:
The current `Paint`. May be `null`.

setShadowBlurRadius

`public void setShadowBlurRadius(float r)`

(continued from last page)

Property: The blur radius for the blur filter. Greater means more smeared shadow but it takes longer to compute.

Parameters:

`r` - The new blur radius. 5 is default.

---

getShadowPlaceRect

`public PlaceRect getShadowPlaceRect()`

Property: How the shadow will be placed relative to the main shape. The `com.miginfocom.util.gfx.geometry.PlaceRect` is an interface that basically takes one `java.awt.Rectangle` and return another one. This makes it very flexible.

Returns:

The current place rect.

See Also:

`com.miginfocom.util.gfx.geometry.AbsRect`
`com.miginfocom.util.gfx.geometry.AlignRect`
`com.miginfocom.util.gfx.geometry.AspectRatioRect`

---

setShadowPlaceRect

`public void setShadowPlaceRect(PlaceRect pr)`

Property: How the shadow will be placed relative to the main shape. The `com.miginfocom.util.gfx.geometry.PlaceRect` is an interface that basically takes one `java.awt.Rectangle` and return another one. This makes it very flexible.

E.g:

```
new AbsRect(new Insets(5, 5, 5, 5));
```

Parameters:

`pr` - The new place rect.

See Also:

`com.miginfocom.util.gfx.geometry.AbsRect`
`com.miginfocom.util.gfx.geometry.AlignRect`
`com.miginfocom.util.gfx.geometry.AspectRatioRect`

---

getShadowCornerRadius

`public double getShadowCornerRadius()`

Property: The corner radius for the shadow `com.miginfocom.util.gfx.RoundRectangle`.

Returns:

The current radius.

See Also:

`setRoundRect(double, double, double, double, double, double)`

(continued from last page)

setShadowCornerRadius

`public void` **`setShadowCornerRadius`**`(double radius)`

Property: The corner radius for the shadow `com.miginfocom.util.gfx.RoundRectangle`.

Parameters:

`radius` - The new radius.

See Also:

`setRoundRect(double, double, double, double, double, double)`

---

getTitleTemplate

`public String` **`getTitleTemplate`**`()`

Property: The template text that will show in the title of the shape. Since the title, or at least parts thereof, is different for every activity but there is only one bean one has to use a template text that will be expanded for every painted activity.

Normally a template will look something like this:

```
"$startTime$-$endTimeExcl$ $description$"
```

however it is better to write it in code something like this:

```
ActivityInteractor.TEMPL_START_TIME.getName() + "-" +
ActivityInteractor.TEMPL_END_TIME_INCL.getName() + " " +
ActivityInteractor.TEMPL_DESCRIPTION.getName();
```

since `ActivityInteractor` is where the special templates is located.

If a special template isn't found the text between the `Interactor.TEMPLATE_CHAR ($)` chars will be converted to a property and fetched from the activity such that:

```
"$assignedTo$"
```

will be converted to:

```
activity.getProperty(PropertyKey.getKey("assignedTo"));
```

This means that any property of the activity can be expanded with this template system.

Default is: `"$startTime$ - $endTimeExcl$ ($timeZoneShort$)"`.

Returns:

The current template string.

See Also:

`com.miginfocom.calendar.activity.ActivityInteractor`
`(for instance)`

---

setTitleTemplate

`public void` **`setTitleTemplate`**`(String s)`

(continued from last page)

Property: The template text that will show in the title of the shape. Since the title, or at least parts thereof, is different for every activity but there is only one bean one has to use a template text that will be expanded for every painted activity.

Normally a template will look something like this:

```
"$startTime$-$endTimeExcl$ $description$"
```

however it is better to write it in code something like this:

```
ActivityInteractor.TEMPL_START_TIME.getName() + "-" +
ActivityInteractor.TEMPL_END_TIME_INCL.getName() + " " +
ActivityInteractor.TEMPL_DESCRIPTION.getName();
```

since `ActivityInteractor` is where the special templates is located.

If a special template isn't found the text between the `Interactor.TEMPLATE_CHAR ($)` chars will be converted to a property and fetched from the activity such that:

```
"$assignedTo$"
```

will be converted to:

```
activity.getProperty(PropertyKey.getKey("assignedTo"));
```

This means that any property of the activity can be expanded with this template system.

Default is: `"$startTime$ - $endTimeExcl$ ($timeZoneShort$)"`.

Parameters:

`s` - The new tempalte string.

See Also:

`com.miginfocom.calendar.activity.ActivityInteractor (for instance)`

---

getTextTemplate

```
public String getTextTemplate()
```

Property: The template text for the main text. See `getTitleTemplate()` for and explanation of the template system

Default is: `"$summary$"`.

Returns:

The current template.

---

setTextTemplate

```
public void setTextTemplate(String s)
```

Property: The template text for the main text. See `getTitleTemplate()` for and explanation of the template system

Default is: `"$summary$"`.

Parameters:

`s` - The new template.

---

getAntiAlias

```
public int getAntiAlias()
```

Property: The anti aliasing hint used when drawing the graphics for the shape. For instance `AA_HINT_ON`.

Returns:

(continued from last page)

The current hint.

See Also:
`confAntiAliasingHint`

---

setAntiAlias

`public void **setAntiAlias**(int hint)`

Property: The anti aliasing hint used when drawing the graphics for the shape. For instance `AA_HINT_ON`.

Parameters:
`hint` - The new hint.

See Also:
`confAntiAliasingHint`

---

getTextAntiAlias

`public int **getTextAntiAlias**()`

Property: The anti aliasing hint used when drawing the text for the shape. For instance `AA_HINT_ON`.

Returns:
The current hint. May be `null`.

See Also:
`confAntiAliasingHint`

---

setTextAntiAlias

`public void **setTextAntiAlias**(int hint)`

Property: The anti aliasing hint used when drawing the text for the shape. For instance `AA_HINT_ON`.

Parameters:
`hint` - The new hint.

See Also:
`confAntiAliasingHint`

---

getTitleForeground

`public java.awt.Paint **getTitleForeground**()`

Property: The `Paint` used to draw the title.

Returns:
The current paint. May be `null`.

---

setTitleForeground

`public void **setTitleForeground**(java.awt.Paint paint)`

Property: The `Paint` used to draw the title.

Parameters:
`paint` - The new paint. May be `null` in which case the title isn't drawn.

getTextForeground

`public java.awt.Paint getTextForeground()`

Property: The `Paint` used to draw the main text.

Returns:
The current paint. May be `null`.

setTextForeground

`public void setTextForeground(java.awt.Paint paint)`

Property: The `Paint` used to draw the main text.

Parameters:
`paint` - The new paint. May be `null` in which case the title isn't drawn.

getOutlinePaint

`public java.awt.Paint getOutlinePaint()`

Property: The `Paint` used to draw the outline.

Returns:
The current paint. May be `null`.

setOutlinePaint

`public void setOutlinePaint(java.awt.Paint paint)`

Property: The `Paint` used to draw the outline.

Parameters:
`paint` - The new paint. May be `null` in which case the outline isn't drawn.

getPlaceRect

`public PlaceRect getPlaceRect()`

Property: How this shape will relate to the reference rectangle (bounds) given by the activity layout system. It can for instance be used to offset it or something else. Norhally though there will just be a 1:1 mapping which can be done with `FILL`.

Returns:
The current place rect. May be `null`.

setPlaceRect

`public void setPlaceRect(PlaceRect placeRect)`

Property: How this shape will relate to the reference rectangle (bounds) given by the activity layout system. It can for instance be used to offset it or something else. Norhally though there will just be a 1:1 mapping which can be done with `FILL`.

Parameters:
`placeRect` - The new place rect. May be `null`, but the behaviour for that is undefined.

getTextPlaceRect

`public PlaceRect getTextPlaceRect()`

Property: How this shape's main text will relate to the reference rectangle (bounds) given by the activity layout system.

Default is: `new AbsRect(new AtStart(2), new AtStart(1), new AtEnd(0), new AtStart(14), null, null, null)`.

Returns:
The current place rect. May be `null`.

setTextPlaceRect

`public void setTextPlaceRect(PlaceRect placeRect)`

Property: How this shape's main text will relate to the reference rectangle (bounds) given by the activity layout system.

Default is: `new AbsRect(new AtStart(2), new AtStart(1), new AtEnd(0), new AtStart(14), null, null, null)`.

Parameters:
`placeRect` - The new place rect. May be `null`, but the behaviour for that is undefined.

getTextAlignX

`public AtRefRangeNumber getTextAlignX()`

Property: Where within the place rect for the text to draw the text string.

Returns:
The alignment for the text. Might be `null`.

See Also:
`getTextPlaceRect()`

setTextAlignX

`public void setTextAlignX(AtRefRangeNumber x)`

Property: Where within the place rect for the text to draw the text string.

Parameters:
`x` - The new alignment for the text. Might be `null`.

See Also:
`getTextPlaceRect()`

getTextAlignY

`public AtRefRangeNumber getTextAlignY()`

Property: Where within the place rect for the text to draw the text string.

Returns:
The alignment for the text. Might be `null`.

See Also:

(continued from last page)

`getTextPlaceRect()`

---

setTextAlignY

```
public void setTextAlignY(AtRefRangeNumber y)
```

Property: Where within the place rect for the text to draw the text string.

Parameters:
`y` - The new alignment for the text. Might be `null`.

See Also:
`getTextPlaceRect()`

---

getTitlePlaceRect

```
public PlaceRect getTitlePlaceRect()
```

Property: How this shape's title will relate to the reference rectangle (bounds) given by the activity layout system.

Default is: `new AbsRect(new AtStart(2), new AtStart(16), new AtEnd(0), new AtEnd(0), null, null, null)`.

Returns:
The current place rect. May be `null`.

---

setTitlePlaceRect

```
public void setTitlePlaceRect(PlaceRect placeRect)
```

Property: How this shape's title will relate to the reference rectangle (bounds) given by the activity layout system.

Default is: `new AbsRect(new AtStart(2), new AtStart(16), new AtEnd(0), new AtEnd(0), null, null, null)`.

Parameters:
`placeRect` - The new place rect. May be `null`, but the behaviour for that is undefined.

---

getTitleAlignX

```
public AtRefRangeNumber getTitleAlignX()
```

Property: Where within the place rect for the title to draw the title text.

Returns:
The alignment for the text. Might be `null`.

See Also:
`getTitlePlaceRect()`

---

setTitleAlignX

```
public void setTitleAlignX(AtRefRangeNumber x)
```

Property: Where within the place rect for the title to draw the title text.

Parameters:
`x` - The new alignment for the text. Might be `null`.

(continued from last page)

See Also:
`setTitlePlaceRect(PlaceRect)`

getTitleAlignY

`public AtRefRangeNumber getTitleAlignY()`

Property: Where within the place rect for the title to draw the title text.

Returns:
The alignment for the text. Might be `null`.

See Also:
`getTitlePlaceRect()`

setTitleAlignY

`public void setTitleAlignY(AtRefRangeNumber y)`

Property: Where within the place rect for the title to draw the title text.

Parameters:
`y` - The new alignment for the text. Might be `null`.

See Also:
`setTitlePlaceRect(PlaceRect)`

getTitleFont

`public java.awt.Font getTitleFont()`

Property: The font used to draw the title in the shape.

Returns:
The current font. May be `null`.

setTitleFont

`public void setTitleFont(java.awt.Font font)`

Property: The font used to draw the title in the shape.

Parameters:
`font` - The new font. May be `null` in which case the title will not be drawn.

getTextFont

`public java.awt.Font getTextFont()`

Property: The font used to draw the main text in the shape.

Returns:
The current font. May be `null`.

setTextFont

`public void setTextFont(java.awt.Font font)`

(continued from last page)

Property: The font used to draw the main text in the shape.

Parameters:
`font` - The new font. May be `null` in which case the title will not be drawn.

getBackground

`public java.awt.Paint getBackground()`

Property: The background paint for the shape.

Returns:
The current background paint. may be `null`.

setBackground

`public void setBackground(java.awt.Paint paint)`

Property: The background paint for the shape.

Parameters:
`paint` - The new background paint. May be `null` in which case it isn't painted.

getOutlineStrokeWidth

`public float getOutlineStrokeWidth()`

Property: The width of the outline around the shape.

Returns:
The current width. Default is 1.

setOutlineStrokeWidth

`public void setOutlineStrokeWidth(float width)`

Property: The width of the outline around the shape.

Parameters:
`width` - The new width.

getCornerRadius

`public double getCornerRadius()`

Property: The corner radius for the outline `com.miginfocom.util.gfx.RoundRectangle`.

Returns:
The current radius.

See Also:
`setRoundRect(double, double, double, double, double, double)`

setCornerRadius

`public void setCornerRadius(double radius)`

Property: The corner radius for the outline `com.miginfocom.util.gfx.RoundRectangle`.

(continued from last page)

Parameters:
`radius` - The new radius.

See Also:
`setRoundRect(double, double, double, double, double, double)`

---

addMouseInteractionListener

`public void` **`addMouseInteractionListener`**`(MouseInteractionListener l)`

Adds a listener that listens to all mouse events on this shape and all sub shapes that has a hit area report set to true.

Note that the listener will be installed on the default AShape which can be used for many date areas. To only handle events originating from a single date area you can do this:

```
if (event.getOriginalEvent().getComponent() == myDateAreaBean.getDateArea()) {
        // Put code to handle the event here...
}
```

Parameters:
`l` - The listener

See Also:
`A_REPORT_HIT_AREA`
`addMouseInteractionListener(MouseInteractionListener, boolean)`

Since:
6.0

---

addMouseInteractionListener

`public void` **`addMouseInteractionListener`**`(MouseInteractionListener l, boolean asWeakRef)`

Adds a listener that listens to all mouse events on this shape and all sub shapes that has a hit area report set to true.

Parameters:
`l` - The listener
`asWeakRef` - If the listener should be added wrapped in a `java.lang.ref.WeakReference`. This defers memory leak problems since the garbage collector can collect the listener if it is only referenced from this list.

Note! This (weak reference) can not be used with listeners that doesn't have another real (a.k.a Strong) reference to it, as for instance an annonymous inner class. If one such listener is added it will be removed almost immediately by the garbage collector.

See Also:
`A_REPORT_HIT_AREA`

Since:
6.0

(continued from last page)

removeMouseInteractionListener

```
public void removeMouseInteractionListener(MouseInteractionListener l)
```

Removes the listener if it exists locally.

Parameters:

`l` - The listener.

Since:

6.0

com.miginfocom.beans
Class CategoryHeaderBean

```
java.lang.Object
   |-com.miginfocom.beans.AbstractBean
      |-com.miginfocom.beans.AbstractHeaderBean
         |-com.miginfocom.beans.CategoryHeaderBean
```

All Implemented Interfaces:
Serializable

Direct Known Subclasses:
WestCategoryHeaderBean, NorthCategoryHeaderBean

---

public abstract class **CategoryHeaderBean**
extends AbstractHeaderBean

A header object that wraps (aggregates) a real `com.miginfocom.calendar.header.Header` implementation.

This bean is for simplifying the usage of a header and to make the header a visual JavaBean.

Note! The category header will end up on the si

Constructor Summary

public	CategoryHeaderBean()

Method Summary

void	addInteractionListener(InteractionListener l) Adds a listener that listens to InteractionEvents.
void	addInteractionListener(InteractionListener l, boolean asWeakRef) Adds a listener that listens to InteractionEvents.
int	getCategoryDepth()
javax.swing.border.Border	getCellBorder() Property: The javax.swing.border.Border used to draw around the cells in the header.
java.awt.Cursor	getCellCursor() Property: The java.awt.Cursor that the mouse pointer should change to when over a cell (and not label).
Integer	getForcedHeaderSize() Property: The size (height if top/bottom and width if left/right) of the header.
Header	getHeader()
DefaultSubRowLevel[]	getHeaderLevels() Property: The specification for the levels that correspond to the category level.

java.awt.Cursor	getKnobCursor() Property: The java.awt.Cursor that the mouse pointer should change to when over +- knob.
XtdImage	getKnobExpandedImage() Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows.
XtdImage	getKnobFoldedImage() Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows.
boolean	getKnobFoldOnPress() Property: If the corresponding sub row should be folded when the user press the knob image in the category header.
AtRefRangeNumber	getKnobImageAlignX() Property: How the knob image should be aligned horizontally.
AtRefRangeNumber	getKnobImageAlignY() Property: How the knob image should be aligned vertically.
java.awt.Cursor	getLabelCursor() Property: The java.awt.Cursor that the mouse pointer should change to when over a category label.
boolean	getLabelFoldOnPress() Property: If the corresponding sub row should be folded when the user press the label in the category header.
boolean	getNoExpandedFolderGridLine() **Deprecated.** *Use DateAreaBean#getNoExpandedFolderGridLine instead.*
XtdImage	getRowFolderImage() Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should represent a folder and are shown for sub rows that aren't leaves.
AtRefRangeNumber	getRowImageAlignX() Property: How the row image should be aligned horizontally.
AtRefRangeNumber	getRowImageAlignY() Property: How the row image should be aligned vertically.
XtdImage	getRowLeafImage() Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should represent a folder and are shown for sub rows that **doesn't** have sub rows.
int	getTextAntiAlias() Property: The anti aliasing hint used when drawing the text for the shape.
void	interactionOccured(InteractionEvent e)
boolean	isFolderOverlapChildren() Property: If the folder category rows, which contains other rows, should get it's secondary dimension's (along the header) size stretched over all its sub rows bounds.
void	removeInteractionListener(InteractionListener l) Removes the listener.

void	setCellBorder(javax.swing.border.Border border) Property: The javax.swing.border.Border used to draw around the cells in the header.
void	setCellCursor(java.awt.Cursor cur) Property: The java.awt.Cursor that the mouse pointer should change to when over a cell (and not label).
boolean	setDateAreaContainer(DateAreaBean container)
void	setFolderOverlapChildren(boolean b) Property: If the folder category rows, which contains other rows, should get it's secondary dimension's (along the header) size stretched over all its sub rows bounds.
void	setForcedHeaderSize(Integer size) Property: The size (height if top/bottom and width if left/right) of the header.
void	setHeaderLevels(DefaultSubRowLevel[] rows) Property: The specification for the levels that correspond to the category level.
void	setKnobCursor(java.awt.Cursor cur) Property: The java.awt.Cursor that the mouse pointer should change to when over +- knob.
void	setKnobExpandedImage(XtdImage image) Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows.
void	setKnobFoldedImage(XtdImage image) Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows.
void	setKnobFoldOnPress(boolean b) Property: If the corresponding sub row should be folded when the user press the knob image in the category header.
void	setKnobImageAlignX(AtRefRangeNumber align) Property: How the knob image should be aligned horizontally.
void	setKnobImageAlignY(AtRefRangeNumber align) Property: How the knob image should be aligned vertically.
void	setLabelCursor(java.awt.Cursor cur) Property: The java.awt.Cursor that the mouse pointer should change to when over a category label.
void	setLabelFoldOnPress(boolean b) Property: If the corresponding sub row should be folded when the user press the label in the category header.
void	setNoExpandedFolderGridLine(boolean b) **Deprecated.** *Use DateAreaBean#setNoExpandedFolderGridLine instead.*
void	setRowFolderImage(XtdImage image) Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should represent a folder and are shown for sub rows that aren't leaves.
void	setRowImageAlignX(AtRefRangeNumber align) Property: How the row image should be aligned horizontally.

void	setRowImageAlignY(AtRefRangeNumber align) Property: How the row image should be aligned vertically.
void	setRowLeafImage(XtdImage image) Property: A com.miginfocom.util.gfx.XtdImage (which is a very flexible Image/Icon) that should represent a folder and are shown for sub rows that **doesn't** have sub rows.
void	setTextAntiAlias(int hint) Property: The anti aliasing hint used when drawing the text for the shape.

Methods inherited from class com.miginfocom.beans.AbstractHeaderBean

getBackgroundPaint, getContainer, getEdge, getExpandToCorner, isVisible, revalidateRepaintContainer, setBackgroundPaint, setDateAreaContainer, setEdge, setExpandToCorner, setVisible

Methods inherited from class com.miginfocom.beans.AbstractBean

addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

CategoryHeaderBean

```
public CategoryHeaderBean()
```

Methods

setDateAreaContainer

```
public boolean setDateAreaContainer(DateAreaBean container)
```

Sets the com.miginfocom.calendar.datearea.DateAreaContainer that this header should decorate. This will be called by the controller that handles them both and the header will not work unless this property has been set.

getHeader

```
public Header getHeader()
```

getCategoryDepth

```
public int getCategoryDepth()
```

(continued from last page)

interactionOccured

`public void interactionOccured(InteractionEvent e)`

getNoExpandedFolderGridLine

`public boolean getNoExpandedFolderGridLine()`

Deprecated. *Use DateAreaBean#getNoExpandedFolderGridLine instead.*

Property: If the grid lines around a folded (collapsed) folder sub row should be merged to one. This is to avoid the double grid lines that would be visible around rows with size 0 otherwise. It is purely a visual setting and won't affect any data or functionality.

Returns:
Always `false`.

setNoExpandedFolderGridLine

`public void setNoExpandedFolderGridLine(boolean b)`

Deprecated. *Use DateAreaBean#setNoExpandedFolderGridLine instead.*

Deprecated because of duplication.

getLabelFoldOnPress

`public boolean getLabelFoldOnPress()`

Property: If the corresponding sub row should be folded when the user press the label in the category header. This is done by catching the `com.miginfocom.ashape.interaction.InteractionEvent` and fold the corresponding row. This can also be done manually, to get more control, by adding a listener with `addInteractionListener(InteractionListener)` and there perform code like:

```
Object interacted = e.getInteractor().getInteracted();
 if (interacted instanceof GridRow) {
    GridRow row = (GridRow) interacted;
    row.setFolded(!row.isFolded());

    revalidateRepaintContainer();
 }
```

Returns:
The current value.

See Also:
`getKnobFoldOnPress()`

(continued from last page)

setLabelFoldOnPress

`public void setLabelFoldOnPress(boolean b)`

Property: If the corresponding sub row should be folded when the user press the label in the category header. This is done by catching the `com.miginfocom.ashape.interaction.InteractionEvent` and fold the corresponding row. This can also be done manually, to get more control, by adding a listener with `addInteractionListener(InteractionListener)` and there perform code like:

```
Object interacted = e.getInteractor().getInteracted();
 if (interacted instanceof GridRow) {
    GridRow row = (GridRow) interacted;
    row.setFolded(!row.isFolded());

    revalidateRepaintContainer();
 }
```

Parameters:

`b` - The new value.

See Also:

`setKnobFoldOnPress(boolean)`

---

getLabelCursor

`public java.awt.Cursor getLabelCursor()`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over a category label.

Returns:

The current Cursor. May be `null` which means no change.

---

setLabelCursor

`public void setLabelCursor(java.awt.Cursor cur)`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over a category label.

Parameters:

`cur` - The new Cursor. May be `null` which means no change.

---

getCellCursor

`public java.awt.Cursor getCellCursor()`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over a cell (and not label).

Returns:

The current Cursor. May be `null` which means no change.

setCellCursor

`public void setCellCursor(java.awt.Cursor cur)`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over a cell (and not label).

Parameters:

`cur` - The new Cursor. May be `null` which means no change.

getKnobFoldOnPress

`public boolean getKnobFoldOnPress()`

Property: If the corresponding sub row should be folded when the user press the knob image in the category header. This is done by catching the `com.miginfocom.ashape.interaction.InteractionEvent` and fold the corresponding row. This can also be done manually, to get more control, by adding a listener with `addInteractionListener(InteractionListener)` and there perform code like:

```
Object interacted = e.getInteractor().getInteracted();
 if (interacted instanceof GridRow) {
    GridRow row = (GridRow) interacted;
    row.setFolded(!row.isFolded());

    revalidateRepaintContainer();
 }
```

Returns:

The current value.

See Also:

`getLabelFoldOnPress()`

setKnobFoldOnPress

`public void setKnobFoldOnPress(boolean b)`

(continued from last page)

Property: If the corresponding sub row should be folded when the user press the knob image in the category header. This is done by catching the `com.miginfocom.ashape.interaction.InteractionEvent` and fold the corresponding row. This can also be done manually, to get more control, by adding a listener with `addInteractionListener(InteractionListener)` and there perform code like:

```
Object interacted = e.getInteractor().getInteracted();
 if (interacted instanceof GridRow) {
    GridRow row = (GridRow) interacted;
    row.setFolded(!row.isFolded());

    revalidateRepaintContainer();
 }
```

Parameters:

`b` - The new value.

See Also:

`setLabelFoldOnPress(boolean)`

getKnobCursor

`public java.awt.Cursor getKnobCursor()`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over +- knob.

Returns:

The current Cursor. May be `null` which means no change.

setKnobCursor

`public void setKnobCursor(java.awt.Cursor cur)`

Property: The `java.awt.Cursor` that the mouse pointer should change to when over +- knob.

Parameters:

`cur` - The new Cursor. May be `null` which means no change.

getRowFolderImage

`public XtdImage getRowFolderImage()`

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should represent a folder and are shown for sub rows that aren't leaves.

Returns:

The current image or `null` if no image should be shown/used.

setRowFolderImage

`public void setRowFolderImage(XtdImage image)`

(continued from last page)

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should represent a folder and are shown for sub rows that aren't leaves.

Parameters:

`image` - The new image or `null` if no image should be shown/used.

---

getRowLeafImage

```
public XtdImage getRowLeafImage()
```

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should represent a folder and are shown for sub rows that **doesn't** have sub rows.

Returns:

The current image or `null` if no image should be shown/used.

---

setRowLeafImage

```
public void setRowLeafImage(XtdImage image)
```

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should represent a folder and are shown for sub rows that **doesn't** have sub rows.

Parameters:

`image` - The new image or `null` if no image should be shown/used.

---

getForcedHeaderSize

```
public Integer getForcedHeaderSize()
```

Property: The size (height if top/bottom and width if left/right) of the header. Normally this value is calculated from the setted `HeaderLevels` (see `setHeaderLevels(com.miginfocom.calendar.header.DefaultSubRowLevel[])`) but sometimes a specific size is needed or is simpler to set.

Depending on the setted `HeaderLevels` those will span this size, but they don't have to.

Returns:

The current size or `null` which means that the header should use the `HeaderLevels` to caclulate the preferred size.

---

setForcedHeaderSize

```
public void setForcedHeaderSize(Integer size)
```

Property: The size (height if top/bottom and width if left/right) of the header. Normally this value is calculated from the setted `HeaderLevels` (see `setHeaderLevels(com.miginfocom.calendar.header.DefaultSubRowLevel[])`) but sometimes a specific size is needed or is simpler to set.

Depending on the setted `HeaderLevels` those will span this size, but they don't have to.

Parameters:

`size` - The new size or `null` which means that the header should use the `HeaderLevels` to caclulate the preferred size.

---

getHeaderLevels

```
public DefaultSubRowLevel[] getHeaderLevels()
```

(continued from last page)

Property: The specification for the levels that correspond to the category level. The main row, which doesn't represent a category but actually the collection of subrows that represents the categories, is level 0. The root category is level 1 and it's direct children level 2 and so on.

This means that a level will get the same attributes normally. It may be background, foreground, color and such things and is outlined in the `com.miginfocom.calendar.header.DefaultSubRowLevel` instances in the array. This also includes mouse over and mouse pressed effects.

Returns:

The current row specs. Never `null`.

See Also:

`setHeaderLevels(com.miginfocom.calendar.header.DefaultSubRowLevel[])`

---

setHeaderLevels

```
public void setHeaderLevels(DefaultSubRowLevel[] rows)
```

(continued from last page)

Property: The specification for the levels that correspond to the category level. The main row, which doesn't represent a category but actually the collection of subrows that represents the categories, is level 0. The root category is level 1 and it's direct children level 2 and so on.

Example 1:

```
Level 0 (main row)
  Personnel (Level 1)
    Chris (Level 2)
    Susan (Level 2)
  Aircraft (Level 1)
    SE001 (Level 2)
    SE002 (Level 2)
    SE002 (Level 2)
<pre>

Example 2:
<pre>
                |   Other, top header...
________________|_________________
|   |   |       |
| P | L | Work  |
| e | i |       |
| r | s |=======|
| s | a | Home  |
| o |   |       |
| n |===|=======|   The date area
| n |   |       |
| e | M | Work  |
| l | a |=======|
|   | r |       |
|   | k | Home  |
________________|_________________
```

When the columns is level 1, 2 and 3. The whole row is level 0.

This means that a level will get the same attributes normally. It may be background, foreground, color and such things and is outlined in the `com.miginfocom.calendar.header.DefaultSubRowLevel` instances in the array. This also includes mouse over and mouse pressed effects.

Parameters:

`rows` - The new rows. `null` is same as empty array.

isFolderOverlapChildren

`public boolean` **`isFolderOverlapChildren`**`()`

Property: If the folder category rows, which contains other rows, should get it's secondary dimension's (along the header) size stretched over all its sub rows bounds. Example 2 in `setHeaderLevels(DefaultSubRowLevel[])` shows how this might look. Note that this is only in the secondary dimension.

Returns:

(continued from last page)

The folder rows' size should include its childrens bounds in the secondary dimension.

setFolderOverlapChildren

```
public void setFolderOverlapChildren(boolean b)
```

Property: If the folder category rows, which contains other rows, should get it's secondary dimension's (along the header) size stretched over all its sub rows bounds. Example 2 in `setHeaderLevels(DefaultSubRowLevel[])` shows how this might look. Note that this is only in the secondary dimension.

Parameters:

`b` - Whether folder rows' size should include its childrens bounds in the secondary dimension.

getCellBorder

```
public javax.swing.border.Border getCellBorder()
```

Property: The `javax.swing.border.Border` used to draw around the cells in the header.

Returns:

The current border or `null` if no border should be painted.

setCellBorder

```
public void setCellBorder(javax.swing.border.Border border)
```

Property: The `javax.swing.border.Border` used to draw around the cells in the header.

Parameters:

`border` - The new border or `null` if no border should be painted.

getKnobExpandedImage

```
public XtdImage getKnobExpandedImage()
```

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows. This is for folder rows that are in their expanded state.

Returns:

The current image or `null` for none.

setKnobExpandedImage

```
public void setKnobExpandedImage(XtdImage image)
```

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows. This is for folder rows that are in their expanded state.

Parameters:

`image` - The new image or `null` for none.

getKnobFoldedImage

```
public XtdImage getKnobFoldedImage()
```

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows. This is for folder rows that are in their folded state.

Returns:

(continued from last page)

The current image or `null` for none.

setKnobFoldedImage

`public void **setKnobFoldedImage**(XtdImage image)`

Property: A `com.miginfocom.util.gfx.XtdImage` (which is a very flexible `Image/Icon`) that should decorate the knob (handle, e.g. +-) that you press th expand/fold the folder rows. This is for folder rows that are in their folded state.

Parameters:

`image` - The new image or `null` for none.

getKnobImageAlignX

`public AtRefRangeNumber **getKnobImageAlignX**()`

Property: How the knob image should be aligned horizontally. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Returns:

The current alignment or `null` for which the behaviour is undefined.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

setKnobImageAlignX

`public void **setKnobImageAlignX**(AtRefRangeNumber align)`

Property: How the knob image should be aligned horizontally. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Parameters:

`align` - The new alignment. Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

getKnobImageAlignY

`public AtRefRangeNumber **getKnobImageAlignY**()`

Property: How the knob image should be aligned vertically. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Returns:

The current alignment. Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

(continued from last page)

setKnobImageAlignY

`public void setKnobImageAlignY(AtRefRangeNumber align)`

Property: How the knob image should be aligned vertically. E.g.: `new AtFraction(0.5f)` for centered or `new AtEnd(-2f)` for two pixels from the bottom edge.

Parameters:

`align` - The new alignment Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

---

getRowImageAlignX

`public AtRefRangeNumber getRowImageAlignX()`

Property: How the row image should be aligned horizontally. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Returns:

The current alignment. Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

---

setRowImageAlignX

`public void setRowImageAlignX(AtRefRangeNumber align)`

Property: How the row image should be aligned horizontally. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Parameters:

`align` - The new alignment. Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

---

getRowImageAlignY

`public AtRefRangeNumber getRowImageAlignY()`

Property: How the row image should be aligned vertically. E.g.: `new AtFraction(0.5f)` for centered or `new AtStart(2f)` for two pixels from the left edge.

Returns:

The current alignment. Not `null`.

See Also:

`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

setRowImageAlignY

`public void setRowImageAlignY(AtRefRangeNumber align)`

Property: How the row image should be aligned vertically. E.g.: `new AtFraction(0.5f)` for centered or `new AtEnd(-2f)` for two pixels from the bottom edge.

Parameters:
`align` - The new alignment. Not `null`.

See Also:
`AtFraction`
`com.miginfocom.util.gfx.geometry.numbers.AtStart`
`com.miginfocom.util.gfx.geometry.numbers.AtEnd`

getTextAntiAlias

`public int getTextAntiAlias()`

Property: The anti aliasing hint used when drawing the text for the shape. For instance `AA_HINT_ON`.

Returns:
The current hint. May be `null`.

See Also:
`confAntiAliasingHint`

setTextAntiAlias

`public void setTextAntiAlias(int hint)`

Property: The anti aliasing hint used when drawing the text for the shape. For instance `AA_HINT_PLATFORM`.

Parameters:
`hint` - The new hint.

See Also:
`confAntiAliasingHint`

addInteractionListener

`public void addInteractionListener(InteractionListener l)`

Adds a listener that listens to `InteractionEvents`. Interaction events are normally fired by the Interaction/Interactor/AbstractInteractionBroker framework, used for instance by the `com.miginfocom.ashape.shapes.AShape` framework.

The interaction events that is fired is when the user presses the knob or the label in the header.

This class is actually a re-dispatcher of these events. It is itself an `com.miginfocom.ashape.interaction.InteractionListener` and registers itself on the interaction observable objects it creates. For instance `com.miginfocom.calendar.activity.view.ActivityViews`.

Parameters:
`l` - The listener to add

(continued from last page)

addInteractionListener

```
public void addInteractionListener(InteractionListener l,
         boolean asWeakRef)
```

Adds a listener that listens to `InteractionEvents`. Interaction events are normally fired by the Interaction/Interactor/AbstractInteractionBroker framework, used for instance by the `com.miginfocom.ashape.shapes.AShape` framework.

The interaction events that is fired is when the user presses the knob or the label in the header.

This class is actually a re-dispatcher of these events. It is itself an `com.miginfocom.ashape.interaction.InteractionListener` and registers itself on the interaction observable objects it creates. For instance `com.miginfocom.calendar.activity.view.ActivityViews`.

Parameters:

`l` - The listener to add

`asWeakRef` - If the listener should be added wrapped in a `java.lang.ref.WeakReference`. This defers memory leak problems since the garbage collector can collect the listener if it is only referenced from this list.

Note! This (weak reference) can not be used with listeners that doesn't have another real (a.k.a Strong) reference to it, as for instance an annonymous inner class. If one such listener is added it will be removed almost immediately by the garbage collector.

---

removeInteractionListener

```
public void removeInteractionListener(InteractionListener l)
```

Removes the listener.

Parameters:

`l` - The listener to remove.

com.miginfocom.beans
Class CategoryTreeBean

```
java.lang.Object
   |-java.awt.Component
      |-java.awt.Container
         |-javax.swing.JComponent
            |-javax.swing.JPanel
               |-com.miginfocom.beans.CategoryTreeBean
```

All Implemented Interfaces:
Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.swing.TransferHandler.HasGetTransferHandler, Serializable, javax.accessibility.Accessible

---

public class **CategoryTreeBean**
extends javax.swing.JPanel

Field Summary

public static final	CHECK_CLICKED The Command Value in for the InteractionEvent that is fired when the label is clicked. Value: **checkClicked**
public static final	CHECK_SELECTED_KEY This is the *default* key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting if the check box is concidered selected or not.
public static final	FOLDER_CHECK_BACKGROUND_SHAPE_NAME Value: **ctChkBb_f**
public static final	FOLDER_CHECK_IMAGE_SHAPE_NAME Value: **ctChkImg_f**
public static final	FOLDER_CHECK_OUTLINE_SHAPE_NAME Value: **ctChkOl_f**
public static final	FOLDER_CONTAINER_SHAPE_NAME Value: **ctBLay_f**
public static final	FOLDER_LABEL_SHAPE_NAME Value: **ctLab_f**
public static final	LABEL_CLICKED The Command Value in for the InteractionEvent that is fired when the label is clicked. Value: **labelClicked**

public static final	LABEL_SELECTED_KEY This is the *default* key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting if the label is concidered selected or not.
public static final	LEAF_CHECK_BACKGROUND_SHAPE_NAME Value: **`ctChkBb_l`**
public static final	LEAF_CHECK_IMAGE_SHAPE_NAME Value: **`ctChkImg_l`**
public static final	LEAF_CHECK_OUTLINE_SHAPE_NAME Value: **`ctChkOl_l`**
public static final	LEAF_CONTAINER_SHAPE_NAME Value: **`ctBLay_l`**
public static final	LEAF_LABEL_SHAPE_NAME Value: **`ctLab_l`**
public static final	ROOT_SHAPE_NAME Value: **`ctRoot`**

Fields inherited from class `javax.swing.JComponent`
`TOOL_TIP_TEXT_KEY`, `UNDEFINED_CONDITION`, `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class `java.awt.Component`
`BOTTOM_ALIGNMENT`, `CENTER_ALIGNMENT`, `LEFT_ALIGNMENT`, `RIGHT_ALIGNMENT`, `TOP_ALIGNMENT`

Fields inherited from interface `java.awt.image.ImageObserver`
`ABORT`, `ALLBITS`, `ERROR`, `FRAMEBITS`, `HEIGHT`, `PROPERTIES`, `SOMEBITS`, `WIDTH`

Constructor Summary

public	CategoryTreeBean()

Method Summary

void	addDecorator(Decorator decorator) Adds a decorator.
void	addDecorators(Collection decorators) Calls `addDecorator(com.miginfocom.calendar.decorators.Decorator)` for every element in the collection.
void	addInteractionListener(InteractionListener l) Adds a listener that listens to `InteractionEvents`.

void	addInteractionListener(InteractionListener l, boolean asWeakRef) Adds a listener that listens to InteractionEvents.
void	addNotify()
int	getAntiAliasHint() Property: The anti alias hint for the label text.
java.awt.Paint	getBackgroundPaint() Sets the background paint.
Category	getCategoryAtLocation(int x, int y) A hit test that returns the Category for a point.
boolean	getCategoryAutoRevalidate() Property: If the categories in the CategoryDepository changes the tree will revalidate itself if there is a category root set.
CategoryViewFilter	getCategoryViewFilter() Property: A filter that can be used to filter out which Categories should be visible in the tree.
static Boolean	getCheckedState(PropertyKey key, Category cat) Returns the checked state of a category.
PropertyKey	getCheckSelectedKey() Property: This is the key used to set in the com.miginfocom.calendar.category.Category a Boolean object denoting on if the check is concidered selected or not.
List	getDecorators() Returns a cloned list with the decorators currently installed in this tree.
DemoDataBean	getDemoDataBean() Property: A reference to a demo data bean that creates demo category data for this bean.
java.awt.Paint	getFolderBackground() Property: The anti alias hint for the label text.
java.awt.Paint	getFolderCheckBackground() Property: The paint to use for the background of the check.
java.awt.Cursor	getFolderCheckCursor() Property: The cursor for when the mouse is over the check.
java.awt.Paint	getFolderCheckForeground() Property: The paint to use for the foreground of the check.
XtdImage	getFolderCheckHalfSelectedIcon() Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.
PlaceRect	getFolderCheckIconPlaceRect() Property: Where in the check shape the icon of the actual check should be placed.
java.awt.Paint	getFolderCheckMouseOverBackground() Property: The paint to use for the background of the check when the mouse is hovering over it.

java.awt.Paint	getFolderCheckMouseOverForeground() Property: The paint to use for the foreground of the check when the mouse is hovering over it.
PlaceRect	getFolderCheckPlaceRect() Property: Where the check outline should be placed within the node.
XtdImage	getFolderCheckSelectedIcon() Property: The selected check icon.
java.awt.Shape	getFolderCheckShape() Property: The shape outline of the check box.
int	getFolderCheckWidth() Property: The width in pixels for the check.
java.awt.Font	getFolderFont() Property: The font used to draw the label.
java.awt.Color	getFolderForeground() Property: The foreground color used to draw the label.
int	getFolderIconTextGap() Property: The gap in pixels between the icon and text.
java.awt.Cursor	getFolderLabelCursor() Property: The hover mouse cursor over the label.
java.awt.Paint	getFolderMouseOverBackground() Property: Background paint when the mouse is hovering over the node.
java.awt.Font	getFolderMouseOverFont() Property: The font when the mouse is hovering over the node.
java.awt.Color	getFolderMouseOverForeground() Property: Foreground paint when the mouse is hovering over the node.
Integer	getFolderMouseOverUnderline() Property: The underline width.
int	getFolderRowHeight() Property: The height in pixels for every row in the tree.
java.awt.Paint	getFolderSelectedBackground() Property: The background if the node label is selected.
java.awt.Font	getFolderSelectedFont() Property: The font if the node label is selected.
java.awt.Color	getFolderSelectedForeground() Property: The foreground if the node label is selected.
Integer	getFolderSelectedUnderline() Property: The underline width if the node label is selected.
Integer	getFolderUnderline() Property: The underline width in pixels.

PropertyKey	getLabelSelectedKey() Property: This is the key used to set in the com.miginfocom.calendar.category.Category a Boolean object denoting on if the label is concidered selected or not.
java.awt.Paint	getLeafBackground() Property: The background paint of the node.
java.awt.Paint	getLeafCheckBackground() Property: The paint to use for the background of the check.
java.awt.Cursor	getLeafCheckCursor() Property: The cursor set when the mouse is hovering over the check.
java.awt.Paint	getLeafCheckForeground() Property: The paint to use for the foreground of the check.
XtdImage	getLeafCheckHalfSelectedIcon() Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.
PlaceRect	getLeafCheckIconPlaceRect() Property: Where in the check shape the icon of the actual check should be placed.
java.awt.Paint	getLeafCheckMouseOverBackground() Property: The background for when the mouse is hovering over the check.
java.awt.Paint	getLeafCheckMouseOverForeground() Property: The foreground for when the mouse is hovering over the check.
PlaceRect	getLeafCheckPlaceRect() Property: Where the check outline should be placed within the node.
XtdImage	getLeafCheckSelectedIcon() Property: The icon/image for when the check is selected.
java.awt.Shape	getLeafCheckShape() Property: The shape outline of the check box.
int	getLeafCheckWidth() Property: The width in pixels for the check.
java.awt.Font	getLeafFont() Property: The font of the label.
java.awt.Color	getLeafForeground() Property: The foreground of the label.
int	getLeafIconTextGap() Property: The gap in pixels between the icon and text.
java.awt.Cursor	getLeafLabelCursor() Property: The cursor for when the mouse is hovering over the label.
java.awt.Paint	getLeafMouseOverBackground() Property: The background for when the mouse is hovering over the label.
java.awt.Font	getLeafMouseOverFont() Property: The font for when the mouse is hovering over the label.

java.awt.Color	getLeafMouseOverForeground() Property: The foreground for when the mouse is hovering over the label.
Integer	getLeafMouseOverUnderline() Property: The underline in pixels for when the mouse is hovering over the label.
int	getLeafRowHeight() Property: The height in pixels for every row in the tree.
java.awt.Paint	getLeafSelectedBackground() Property: The background if the node label is selected.
java.awt.Font	getLeafSelectedFont() Property: The font if the node label is selected.
java.awt.Color	getLeafSelectedForeground() Property: The foreground if the node label is selected.
Integer	getLeafSelectedUnderline() Property: The underline in pixels if the node label is selected.
Integer	getLeafUnderline() Property: The underline width in pixels.
java.awt.Dimension	getPreferredSize()
Object	getRootCategoryId() Returns the category ID of the root node.
javax.swing.JScrollPane	getScrollPane() Returns the scroll panel that is used to show the category tree.
boolean	getShowsRootHandles() Property: If the root handles should be visible.
javax.swing.JTree	getTree() Returns the JTree that is the backing component for this tree bean.
void	interactionOccured(InteractionEvent e)
boolean	isFolderCheckSelectable() Property: If the check can be selected.
boolean	isFolderCheckVisible() Property: If the check box is visible.
boolean	isFolderLabelSelectable() Property: If the label should be selectable (press-able)
boolean	isFolderRouteLabelClickToCheck() Property: If a click on the label should be transformed to a click on the check.
boolean	isIgnoreInteractionEvents() Returns if events are currently ignored.
boolean	isLeafCheckSelectable() Property: If the check should be selectable.

boolean	isLeafCheckVisible() Property: If the check box is visible.
boolean	isLeafLabelSelectable() Property: If the label should be selectable.
boolean	isLeafRouteLabelClickToCheck() Property: If a click on the label should be transformed to a click on the check.
boolean	isRootVisible() Property: If the root node should be visible.
boolean	removeDecorator(Class type, boolean inclSubClasses) Removes the first decorator found with the class type `type`, including sub types id `inclSubClasses == true`.
void	removeDecorator(Decorator decorator) Removes a decorator.
void	removeDecorators() Removes all decorators.
void	removeDecorators(Collection decorators) Removes all decorators
void	removeInteractionListener(InteractionListener l) Removes the listener if it is added.
void	removeNotify()
void	revalidateNodes(Object fromCatId, boolean keepExpanded) Revalidate the nodes starting from `fromCatId`.
void	setAntiAliasHint(int hint) Property: The anti alias hint for the label text.
void	setBackgroundPaint(java.awt.Paint bgPaint) Sets the background paint.
void	setCategoryAutoRevalidate(boolean b) Property: If the categories in the `CategoryDepository` changes the tree will revalidate itself if there is a category root set.
void	setCategoryViewFilter(CategoryViewFilter filter) Property: A filter that can be used to filter out which Categories should be visible in the tree.
static void	setCheckedState(PropertyKey key, Category cat, boolean b) Set the selected state for the category.
void	setCheckSelectedKey(PropertyKey key) Property: This is the key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting on if the check is concidered selected or not.
void	setDecorators(Collection decorators) Sets the decorators to use.
void	setDemoDataBean(DemoDataBean b) Property: A reference to a demo data bean that creates demo category data for this bean.

void	setFolderBackground(java.awt.Paint p) Property: The anti alias hint for the label text.
void	setFolderCheckBackground(java.awt.Paint p) Property: The paint to use for the background of the check.
void	setFolderCheckCursor(java.awt.Cursor cur) Property: The cursor for when the mouse is over the check.
void	setFolderCheckForeground(java.awt.Paint p) Property: The paint to use for the foreground of the check.
void	setFolderCheckHalfSelectedIcon(XtdImage image) Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.
void	setFolderCheckIconPlaceRect(PlaceRect r) Property: Where in the check shape the icon of the actual check should be placed.
void	setFolderCheckMouseOverBackground(java.awt.Paint p) Property: The paint to use for the background of the check when the mouse is hovering over it.
void	setFolderCheckMouseOverForeground(java.awt.Paint p) Property: The paint to use for the foreground of the check when the mouse is hovering over it.
void	setFolderCheckPlaceRect(PlaceRect r) Property: Where the check outline should be placed within the node.
void	setFolderCheckSelectable(boolean b) Property: If the check can be selected.
void	setFolderCheckSelectedIcon(XtdImage image) Property: The selected check icon.
void	setFolderCheckShape(java.awt.Shape s) Property: The shape outline of the check box.
void	setFolderCheckWidth(int w) Property: The width in pixels for the check.
void	setFolderCheckVisible(boolean b) Property: If the check box is visible.
void	setFolderFont(java.awt.Font f) Property: The font used to draw the label.
void	setFolderForeground(java.awt.Color c) Property: The foreground color used to draw the label.
void	setFolderIconTextGap(int gap) Property: The gap in pixels between the icon and text.
void	setFolderLabelCursor(java.awt.Cursor cur) Property: The hover mouse cursor over the label.
void	setFolderLabelSelectable(boolean b) Property: If the label should be selectable (press-able)

void	setFolderMouseOverBackground(java.awt.Paint p) Property: Background paint when the mouse is hovering over the node.
void	setFolderMouseOverFont(java.awt.Font f) Property: The font when the mouse is hovering over the node.
void	setFolderMouseOverForeground(java.awt.Color c) Property: Foreground paint when the mouse is hovering over the node.
void	setFolderMouseOverUnderline(Integer b) Property: The underline width.
void	setFolderRouteLabelClickToCheck(boolean b) Property: If a click on the label should be transformed to a click on the check.
void	setFolderRowHeight(int height) Property: The height in pixels for every row in the tree.
void	setFolderSelectedBackground(java.awt.Paint p) Property: The background if the node label is selected.
void	setFolderSelectedFont(java.awt.Font f) Property: The font if the node label is selected.
void	setFolderSelectedForeground(java.awt.Color c) Property: The foreground if the node label is selected.
void	setFolderSelectedUnderline(Integer width) Property: The underline width if the node label is selected.
void	setFolderUnderline(Integer b) Property: The underline width in pixels.
boolean	setIgnoreInteractionEvents(boolean b) Sets if events should be ignored, and thus not fired.
void	setLabelSelectedKey(PropertyKey key) Property: This is the key used to set in the com.miginfocom.calendar.category.Category a Boolean object denoting on if the label is concidered selected or not.
void	setLeafBackground(java.awt.Paint p) Property: The background paint of the node.
void	setLeafCheckBackground(java.awt.Paint p) Property: The paint to use for the background of the check.
void	setLeafCheckCursor(java.awt.Cursor cur) Property: The cursor set when the mouse is hovering over the check.
void	setLeafCheckForeground(java.awt.Paint p) Property: The paint to use for the foreground of the check.
void	setLeafCheckHalfSelectedIcon(XtdImage image) Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.
void	setLeafCheckIconPlaceRect(PlaceRect r) Property: Where in the check shape the icon of the actual check should be placed.

void	setLeafCheckMouseOverBackground(java.awt.Paint p) Property: The background for when the mouse is hovering over the check.
void	setLeafCheckMouseOverForeground(java.awt.Paint p) Property: The foreground for when the mouse is hovering over the check.
void	setLeafCheckPlaceRect(PlaceRect r) Property: Where the check outline should be placed within the node.
void	setLeafCheckSelectable(boolean b) Property: If the check should be selectable.
void	setLeafCheckSelectedIcon(XtdImage image) Property: The icon/image for when the check is selected.
void	setLeafCheckShape(java.awt.Shape s) Property: The shape outline of the check box.
void	setLeafCheckWidth(int w) Property: The width in pixels for the check.
void	setLeafCheckVisible(boolean b) Property: If the check box is visible.
void	setLeafFont(java.awt.Font f) Property: The font of the label.
void	setLeafForeground(java.awt.Color c) Property: The foreground of the label.
void	setLeafIconTextGap(int gap) Property: The gap in pixels between the icon and text.
void	setLeafLabelCursor(java.awt.Cursor cur) Property: The cursor for when the mouse is hovering over the label.
void	setLeafLabelSelectable(boolean b) Property: If the label should be selectable.
void	setLeafMouseOverBackground(java.awt.Paint p) Property: The background for when the mouse is hovering over the label.
void	setLeafMouseOverFont(java.awt.Font f) Property: The font for when the mouse is hovering over the label.
void	setLeafMouseOverForeground(java.awt.Color c) Property: The foreground for when the mouse is hovering over the label.
void	setLeafMouseOverUnderline(Integer b) Property: The underline in pixels for when the mouse is hovering over the label.
void	setLeafRouteLabelClickToCheck(boolean b) Property: If a click on the label should be transformed to a click on the check.
void	setLeafRowHeight(int height) Property: The height in pixels for every row in the tree.
void	setLeafSelectedBackground(java.awt.Paint p) Property: The background if the node label is selected.

void	setLeafSelectedFont(java.awt.Font f) Property: The font if the node label is selected.
void	setLeafSelectedForeground(java.awt.Color c) Property: The foreground if the node label is selected.
void	setLeafSelectedUnderline(Integer width) Property: The underline in pixels if the node label is selected.
void	setLeafUnderline(Integer b) Property: The underline width in pixels.
void	setPreferredSize(java.awt.Dimension preferredSize)
void	setRootCategoryId(Object catIDRoot) Sets the category ID of the root node.
void	setRootVisible(boolean b) Property: If the root node should be visible.
void	setShowsRootHandles(boolean b) Property: If the root handles should be visible.
void	sortDecorators() Resort the decorators that this tree handles.
static void	toggleCheckedState(PropertyKey key, Category cat) Toggles the checked state on the category.

Methods inherited from class `javax.swing.JPanel`

`getAccessibleContext, getUI, getUIClassID, setUI, updateUI`

Methods inherited from class `javax.swing.JComponent`

addAncestorListener, addNotify, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, getAccessibleContext, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getUIClassID, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getWidth, getVisibleRect, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintImmediately, paintImmediately, print, printAll, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update, updateUI

Methods inherited from class `java.awt.Container`

add, add, add, add, add, addContainerListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate

Methods inherited from class `java.awt.Component`

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, deliverEvent, disable, dispatchEvent, doLayout, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getAccessibleContext, getAlignmentX, getAlignmentY, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentAt, getComponentAt, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeys, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getListeners, getLocale, getLocation, getLocation, getLocationOnScreen, getMaximumSize, getMinimumSize, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, invalidate, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusCycleRoot, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, layout, list, list, list, list, list, locate, location, lostFocus, minimumSize, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paint, paintAll, postEvent, preferredSize, prepareImage, prepareImage, print, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeys, setFocusTraversalKeysEnabled, setFont, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusBackward, transferFocusUpCycle, update, validate

Methods inherited from class `java.lang.Object`

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface `java.awt.image.ImageObserver`

imageUpdate

Methods inherited from interface `java.awt.MenuContainer`

getFont, postEvent, remove

Methods inherited from interface `javax.swing.TransferHandler.HasGetTransferHandler`

getTransferHandler

Methods inherited from interface `javax.accessibility.Accessible`

`getAccessibleContext`

Fields

LABEL_SELECTED_KEY

`public static final PropertyKey` **`LABEL_SELECTED_KEY`**

This is the *default* key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting if the label is concidered selected or not. It can also be set on a tree by tree basis using `setLabelSelectedKey(com.miginfocom.util.PropertyKey)`.

CHECK_SELECTED_KEY

`public static final PropertyKey` **`CHECK_SELECTED_KEY`**

This is the *default* key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting if the check box is concidered selected or not. It can also be set on a tree by tree basis using `setCheckSelectedKey(com.miginfocom.util.PropertyKey)`.

LABEL_CLICKED

`public static final java.lang.String` **`LABEL_CLICKED`**

The Command Value in for the InteractionEvent that is fired when the label is clicked.
Constant value: **`labelClicked`**

CHECK_CLICKED

`public static final java.lang.String` **`CHECK_CLICKED`**

The Command Value in for the InteractionEvent that is fired when the label is clicked.
Constant value: **`checkClicked`**

ROOT_SHAPE_NAME

`public static final java.lang.String` **`ROOT_SHAPE_NAME`**

Constant value: **`ctRoot`**

FOLDER_CHECK_BACKGROUND_SHAPE_NAME

`public static final java.lang.String` **`FOLDER_CHECK_BACKGROUND_SHAPE_NAME`**

Constant value: **`ctChkBb_f`**

FOLDER_CHECK_OUTLINE_SHAPE_NAME

`public static final java.lang.String` **`FOLDER_CHECK_OUTLINE_SHAPE_NAME`**

Constant value: **`ctChkOl_f`**

(continued from last page)

FOLDER_CHECK_IMAGE_SHAPE_NAME

`public static final java.lang.String` **`FOLDER_CHECK_IMAGE_SHAPE_NAME`**

Constant value: **`ctChkImg_f`**

FOLDER_LABEL_SHAPE_NAME

`public static final java.lang.String` **`FOLDER_LABEL_SHAPE_NAME`**

Constant value: **`ctLab_f`**

FOLDER_CONTAINER_SHAPE_NAME

`public static final java.lang.String` **`FOLDER_CONTAINER_SHAPE_NAME`**

Constant value: **`ctBLay_f`**

LEAF_CHECK_BACKGROUND_SHAPE_NAME

`public static final java.lang.String` **`LEAF_CHECK_BACKGROUND_SHAPE_NAME`**

Constant value: **`ctChkBb_l`**

LEAF_CHECK_OUTLINE_SHAPE_NAME

`public static final java.lang.String` **`LEAF_CHECK_OUTLINE_SHAPE_NAME`**

Constant value: **`ctChkOl_l`**

LEAF_CHECK_IMAGE_SHAPE_NAME

`public static final java.lang.String` **`LEAF_CHECK_IMAGE_SHAPE_NAME`**

Constant value: **`ctChkImg_l`**

LEAF_LABEL_SHAPE_NAME

`public static final java.lang.String` **`LEAF_LABEL_SHAPE_NAME`**

Constant value: **`ctLab_l`**

LEAF_CONTAINER_SHAPE_NAME

`public static final java.lang.String` **`LEAF_CONTAINER_SHAPE_NAME`**

Constant value: **`ctBLay_l`**

Constructors

(continued from last page)

CategoryTreeBean

```
public CategoryTreeBean()
```

Methods

addNotify

```
public void addNotify()
```

---

removeNotify

```
public void removeNotify()
```

---

getPreferredSize

```
public java.awt.Dimension getPreferredSize()
```

---

setPreferredSize

```
public void setPreferredSize(java.awt.Dimension preferredSize)
```

---

getTree

```
public javax.swing.JTree getTree()
```

Returns the JTree that is the backing component for this tree bean.

Returns:

The JTree that is the backing component for this tree bean.

---

getCategoryAtLocation

```
public Category getCategoryAtLocation(int x,
          int y)
```

A hit test that returns the Category for a point.

Parameters:

`x` - The x value that is relative to the contained tree.
`y` - The y value that is relative to the contained tree.

Returns:

The Category or `null` if no hit.

(continued from last page)

getScrollPane

```
public javax.swing.JScrollPane getScrollPane()
```

Returns the scroll panel that is used to show the category tree.

Returns:
The scroll panel that is used to show the category tree. Never `null`.

---

getCategoryViewFilter

```
public CategoryViewFilter getCategoryViewFilter()
```

Property: A filter that can be used to filter out which Categories should be visible in the tree.

Returns:
The current filter. `null` if no filter.

---

setCategoryViewFilter

```
public void setCategoryViewFilter(CategoryViewFilter filter)
```

Property: A filter that can be used to filter out which Categories should be visible in the tree.

Parameters:
`filter` - The new filter. `null` if no filter.

---

getBackgroundPaint

```
public java.awt.Paint getBackgroundPaint()
```

Sets the background paint. If `!= null` this paint is drawn in the background (if opaque) rather than the normal background color. If this is `null` the normal color retrieved from `Component.getBackground()` is painted instead. If not opaque none of the colors will be painted.

Returns:
The background paint. May be `null`.

---

setBackgroundPaint

```
public void setBackgroundPaint(java.awt.Paint bgPaint)
```

Sets the background paint. If `!= null` this paint is drawn in the background (if opaque) rather than the normal background color. If this is `null` the normal color retrieved from `Component.getBackground()` is painted instead. If not opaque none of the colors will be painted.

Parameters:
`bgPaint` - The background paint. May be `null`.

---

getCategoryAutoRevalidate

```
public boolean getCategoryAutoRevalidate()
```

Property: If the categories in the `CategoryDepository` changes the tree will revalidate itself if there is a category root set.

Returns:
If the auto revalidation is on.

setCategoryAutoRevalidate

`public void setCategoryAutoRevalidate(boolean b)`

Property: If the categories in the `CategoryDepository` changes the tree will revalidate itself if there is a category root set.

Parameters:

`b` - If the auto revalidation is on.

getLabelSelectedKey

`public PropertyKey getLabelSelectedKey()`

Property: This is the key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting on if the label is concidered selected or not.

The key defaults to `LABEL_SELECTED_KEY` but using that means that all category trees will have the same model state for a category, which is sometime wanted and sometimes not.

Returns:

The current key. Never `null`.

setLabelSelectedKey

`public void setLabelSelectedKey(PropertyKey key)`

Property: This is the key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting on if the label is concidered selected or not.

The key defaults to `LABEL_SELECTED_KEY` but using that means that all category trees will have the same model state for a category, which is sometime wanted and sometimes not.

Parameters:

`key` - The new key. `null` will reset it to the default value.

getCheckSelectedKey

`public PropertyKey getCheckSelectedKey()`

Property: This is the key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting on if the check is concidered selected or not.

The key defaults to `CHECK_SELECTED_KEY` but using that means that all category trees will have the same model state for a category, which is sometime wanted and sometimes not.

Returns:

The current key. Never `null`.

setCheckSelectedKey

`public void setCheckSelectedKey(PropertyKey key)`

Property: This is the key used to set in the `com.miginfocom.calendar.category.Category` a `Boolean` object denoting on if the check is concidered selected or not.

The key defaults to `CHECK_SELECTED_KEY` but using that means that all category trees will have the same model state for a category, which is sometime wanted and sometimes not.

Parameters:

(continued from last page)

`key` - The new key. `null` will reset it to the default value.

getDemoDataBean

`public DemoDataBean getDemoDataBean()`

Property: A reference to a demo data bean that creates demo category data for this bean. This is normally only used for testing purposes and should not be used in any production environment.

Returns:
The current value of the property or `null`.

setDemoDataBean

`public void setDemoDataBean(DemoDataBean b)`

Property: A reference to a demo data bean that creates demo category data for this bean. This is normally only used for testing purposes and should not be used in any production environment.

Parameters:
`b` - The new value for the property. May be `null`.

isLeafRouteLabelClickToCheck

`public boolean isLeafRouteLabelClickToCheck()`

Property: If a click on the label should be transformed to a click on the check.

Returns:
The current value. Default is `false`.

setLeafRouteLabelClickToCheck

`public void setLeafRouteLabelClickToCheck(boolean b)`

Property: If a click on the label should be transformed to a click on the check.

Parameters:
`b` - The new value. Default is `false`.

isFolderRouteLabelClickToCheck

`public boolean isFolderRouteLabelClickToCheck()`

Property: If a click on the label should be transformed to a click on the check.

Returns:
The current value. Default is `false`.

setFolderRouteLabelClickToCheck

`public void setFolderRouteLabelClickToCheck(boolean b)`

Property: If a click on the label should be transformed to a click on the check.

Parameters:
`b` - The new value. Default is `false`.

(continued from last page)

isLeafCheckVisible

`public boolean isLeafCheckVisible()`

Property: If the check box is visible.

Returns:
The current value. Default is `true`.

---

setLeafCheckVisible

`public void setLeafCheckVisible(boolean b)`

Property: If the check box is visible.

Parameters:
`b` - The new value. Default is `true`.

---

isFolderCheckVisible

`public boolean isFolderCheckVisible()`

Property: If the check box is visible.

Returns:
The current value. Default is `true`.

---

setFolderCheckVisible

`public void setFolderCheckVisible(boolean b)`

Property: If the check box is visible.

Parameters:
`b` - The new value. Default is `true`.

---

getLeafRowHeight

`public int getLeafRowHeight()`

Property: The height in pixels for every row in the tree.

Returns:
The current value. Default is `17`.

---

setLeafRowHeight

`public void setLeafRowHeight(int height)`

Property: The height in pixels for every row in the tree.

Parameters:
`height` - The new value. Default is `17`.

---

getFolderRowHeight

`public int getFolderRowHeight()`

(continued from last page)

Property: The height in pixels for every row in the tree.

Returns:
The current value. Default is `17`.

setFolderRowHeight

`public void setFolderRowHeight(int height)`

Property: The height in pixels for every row in the tree.

Parameters:
`height` - The new value. Default is `17`.

getShowsRootHandles

`public boolean getShowsRootHandles()`

Property: If the root handles should be visible.

Returns:
The current value. Default is `true`.

setShowsRootHandles

`public void setShowsRootHandles(boolean b)`

Property: If the root handles should be visible.

Parameters:
`b` - The new value. Default is `true`.

getLeafCheckShape

`public java.awt.Shape getLeafCheckShape()`

Property: The shape outline of the check box.

Returns:
The current value. Default is `new RoundRectangle(0, 0, 14, 14, 6, 6)`.

setLeafCheckShape

`public void setLeafCheckShape(java.awt.Shape s)`

Property: The shape outline of the check box.

Parameters:
`s` - The new value. Default is `new RoundRectangle(0, 0, 14, 14, 6, 6)`.

getFolderCheckShape

`public java.awt.Shape getFolderCheckShape()`

Property: The shape outline of the check box.

Returns:
The current value. Default is `new RoundRectangle(0, 0, 14, 14, 6, 6)`.

setFolderCheckShape

`public void setFolderCheckShape(java.awt.Shape s)`

Property: The shape outline of the check box.

Parameters:
`s` - The new value. Default is `new RoundRectangle(0, 0, 14, 14, 6, 6)`.

getLeafCheckIconPlaceRect

`public PlaceRect getLeafCheckIconPlaceRect()`

Property: Where in the check shape the icon of the actual check should be placed.

Returns:
The current value. Default is `new AlignRect(new AtFraction(0.5f), new AtFraction(0.5f))`.

setLeafCheckIconPlaceRect

`public void setLeafCheckIconPlaceRect(PlaceRect r)`

Property: Where in the check shape the icon of the actual check should be placed.

Parameters:
`r` - The new value. Default is `new AlignRect(new AtFraction(0.5f), new AtFraction(0.5f))`.

getFolderCheckIconPlaceRect

`public PlaceRect getFolderCheckIconPlaceRect()`

Property: Where in the check shape the icon of the actual check should be placed.

Returns:
The current value. Default is `new AlignRect(new AtFraction(0.5f), new AtFraction(0.5f))`.

setFolderCheckIconPlaceRect

`public void setFolderCheckIconPlaceRect(PlaceRect r)`

Property: Where in the check shape the icon of the actual check should be placed.

Parameters:
`r` - The new value. Default is `new AlignRect(new AtFraction(0.5f), new AtFraction(0.5f))`.

getLeafCheckPlaceRect

`public PlaceRect getLeafCheckPlaceRect()`

Property: Where the check outline should be placed within the node.

Returns:
The current value. Default is `new AbsRect(new Insets(1, 1, 1, 1))`.

setLeafCheckPlaceRect

`public void setLeafCheckPlaceRect(PlaceRect r)`

(continued from last page)

Property: Where the check outline should be placed within the node.

Parameters:
`r` - The new value. Default is `new AbsRect(new Insets(1, 1, 1, 1))`.

getFolderCheckPlaceRect

`public PlaceRect getFolderCheckPlaceRect()`

Property: Where the check outline should be placed within the node.

Returns:
The current value. Default is `new AbsRect(new Insets(1, 1, 1, 1))`.

setFolderCheckPlaceRect

`public void setFolderCheckPlaceRect(PlaceRect r)`

Property: Where the check outline should be placed within the node.

Parameters:
`r` - The new value. Default is `new AbsRect(new Insets(1, 1, 1, 1))`.

getLeafCheckWidth

`public int getLeafCheckWidth()`

Property: The width in pixels for the check.

Returns:
The current value. Default is `17`.

setLeafCheckWidth

`public void setLeafCheckWidth(int w)`

Property: The width in pixels for the check.

Parameters:
`w` - The new value. Default is `17`.

getFolderCheckWidth

`public int getFolderCheckWidth()`

Property: The width in pixels for the check.

Returns:
The current value. Default is `17`.

setFolderCheckWidth

`public void setFolderCheckWidth(int w)`

Property: The width in pixels for the check.

Parameters:
`w` - The new value. Default is `17`.

getFolderIconTextGap

`public int getFolderIconTextGap()`

Property: The gap in pixels between the icon and text.

Returns:
The current value. Default is `3`.

setFolderIconTextGap

`public void setFolderIconTextGap(int gap)`

Property: The gap in pixels between the icon and text.

Parameters:
`gap` - The new value. Default is `3`.

getLeafIconTextGap

`public int getLeafIconTextGap()`

Property: The gap in pixels between the icon and text.

Returns:
The current value. Default is `3`.

setLeafIconTextGap

`public void setLeafIconTextGap(int gap)`

Property: The gap in pixels between the icon and text.

Parameters:
`gap` - The new value. Default is `3`.

getAntiAliasHint

`public int getAntiAliasHint()`

Property: The anti alias hint for the label text.

Returns:
The current value. Default is `AA_HINT_PLATFORM`.

setAntiAliasHint

`public void setAntiAliasHint(int hint)`

Property: The anti alias hint for the label text.

Parameters:
`hint` - The new value. Default is `AA_HINT_PLATFORM`.

getFolderBackground

`public java.awt.Paint getFolderBackground()`

(continued from last page)

Property: The anti alias hint for the label text.

Returns:
The current value. Default is `AA_HINT_PLATFORM`.

setFolderBackground

```
public void setFolderBackground(java.awt.Paint p)
```

Property: The anti alias hint for the label text.

Parameters:
`p` - The new value. Default is `AA_HINT_PLATFORM`.

getFolderFont

```
public java.awt.Font getFolderFont()
```

Property: The font used to draw the label.

Returns:
The current value. Default is the font for the bean itself. Never `null`.

setFolderFont

```
public void setFolderFont(java.awt.Font f)
```

Property: The font used to draw the label.

Parameters:
`f` - The new value. If `null` set to the font for the bean itself.

getFolderForeground

```
public java.awt.Color getFolderForeground()
```

Property: The foreground color used to draw the label.

Returns:
The current value. Default is the foreground for the bean itself. Never `null`.

setFolderForeground

```
public void setFolderForeground(java.awt.Color c)
```

Property: The foreground color used to draw the label.

Parameters:
`c` - The new value. If `null` set to the foreground for the bean itself.

getFolderCheckCursor

```
public java.awt.Cursor getFolderCheckCursor()
```

Property: The cursor for when the mouse is over the check.

Returns:
The current value. Default is `null`.

setFolderCheckCursor

`public void setFolderCheckCursor(java.awt.Cursor cur)`

Property: The cursor for when the mouse is over the check.

Parameters:
`cur` - The new value. Default is `null`.

getFolderCheckHalfSelectedIcon

`public XtdImage getFolderCheckHalfSelectedIcon()`

Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.

Returns:
The current value. Default is `HALF_CHECK_IMG` (private)

setFolderCheckHalfSelectedIcon

`public void setFolderCheckHalfSelectedIcon(XtdImage image)`

Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.

Parameters:
`image` - The new value. Default is `HALF_CHECK_IMG` (private)

getFolderCheckMouseOverBackground

`public java.awt.Paint getFolderCheckMouseOverBackground()`

Property: The paint to use for the background of the check when the mouse is hovering over it.

Returns:
The current value. Default is `null` which means no mouse over effect.

setFolderCheckMouseOverBackground

`public void setFolderCheckMouseOverBackground(java.awt.Paint p)`

Property: The paint to use for the background of the check when the mouse is hovering over it.

Parameters:
`p` - The new value. Default is `null` which means no mouse over effect.

getFolderCheckMouseOverForeground

`public java.awt.Paint getFolderCheckMouseOverForeground()`

Property: The paint to use for the foreground of the check when the mouse is hovering over it.

Returns:
The current value. Default is `null` which means no mouse over effect.

setFolderCheckMouseOverForeground

`public void setFolderCheckMouseOverForeground(java.awt.Paint p)`

(continued from last page)

Property: The paint to use for the foreground of the check when the mouse is hovering over it.

Parameters:
p - The new value. Default is `null` which means no mouse over effect.

getLeafCheckForeground

`public java.awt.Paint getLeafCheckForeground()`

Property: The paint to use for the foreground of the check.

Returns:
The current value. Default is the foreground of the bean.

setLeafCheckForeground

`public void setLeafCheckForeground(java.awt.Paint p)`

Property: The paint to use for the foreground of the check.

Parameters:
p - The new value. If `null` it is reset to the foreground of the bean.

getFolderCheckForeground

`public java.awt.Paint getFolderCheckForeground()`

Property: The paint to use for the foreground of the check.

Returns:
The current value. Default is the foreground of the bean.

setFolderCheckForeground

`public void setFolderCheckForeground(java.awt.Paint p)`

Property: The paint to use for the foreground of the check.

Parameters:
p - The new value. If `null` it is reset to the foreground of the bean.

getLeafCheckBackground

`public java.awt.Paint getLeafCheckBackground()`

Property: The paint to use for the background of the check.

Returns:
The current value. Default is the background of the bean.

setLeafCheckBackground

`public void setLeafCheckBackground(java.awt.Paint p)`

Property: The paint to use for the background of the check.

Parameters:
p - The new value. If `null` it is reset to the background of the bean.

getFolderCheckBackground

```
public java.awt.Paint getFolderCheckBackground()
```

Property: The paint to use for the background of the check.

Returns:
The current value. Default is the background of the bean.

setFolderCheckBackground

```
public void setFolderCheckBackground(java.awt.Paint p)
```

Property: The paint to use for the background of the check.

Parameters:
`p` - The new value. If `null` it is reset to the background of the bean.

isFolderCheckSelectable

```
public boolean isFolderCheckSelectable()
```

Property: If the check can be selected.

Returns:
The current value. Default is `true`.

setFolderCheckSelectable

```
public void setFolderCheckSelectable(boolean b)
```

Property: If the check can be selected.

Parameters:
`b` - The new value. Default is `true`.

getFolderCheckSelectedIcon

```
public XtdImage getFolderCheckSelectedIcon()
```

Property: The selected check icon.

Returns:
The current value. Default is `CHECK_IMG`

setFolderCheckSelectedIcon

```
public void setFolderCheckSelectedIcon(XtdImage image)
```

Property: The selected check icon.

Parameters:
`image` - The new value. Default is `CHECK_IMG`

getFolderLabelCursor

```
public java.awt.Cursor getFolderLabelCursor()
```

(continued from last page)

Property: The hover mouse cursor over the label.

Returns:
The current value. Default is `null`.

setFolderLabelCursor

```
public void setFolderLabelCursor(java.awt.Cursor cur)
```

Property: The hover mouse cursor over the label.

Parameters:
`cur` - The new value. Default is `null`.

isFolderLabelSelectable

```
public boolean isFolderLabelSelectable()
```

Property: If the label should be selectable (press-able)

Returns:
The current value. Default is `false`.

setFolderLabelSelectable

```
public void setFolderLabelSelectable(boolean b)
```

Property: If the label should be selectable (press-able)

Parameters:
`b` - The new value. Default is `false`.

getFolderMouseOverBackground

```
public java.awt.Paint getFolderMouseOverBackground()
```

Property: Background paint when the mouse is hovering over the node.

Returns:
The current value. Default is `null`.

setFolderMouseOverBackground

```
public void setFolderMouseOverBackground(java.awt.Paint p)
```

Property: Background paint when the mouse is hovering over the node.

Parameters:
`p` - The new value. Default is `null`.

getFolderMouseOverFont

```
public java.awt.Font getFolderMouseOverFont()
```

Property: The font when the mouse is hovering over the node.

Returns:
The current value. Default is `null` which means no special font.

setFolderMouseOverFont

`public void setFolderMouseOverFont(java.awt.Font f)`

Property: The font when the mouse is hovering over the node.

Parameters:
`f` - The new value. Default is `null` which means no special font.

getFolderMouseOverForeground

`public java.awt.Color getFolderMouseOverForeground()`

Property: Foreground paint when the mouse is hovering over the node.

Returns:
The current value. Default is `null`.

setFolderMouseOverForeground

`public void setFolderMouseOverForeground(java.awt.Color c)`

Property: Foreground paint when the mouse is hovering over the node.

Parameters:
`c` - The new value. Default is `null`.

getFolderMouseOverUnderline

`public Integer getFolderMouseOverUnderline()`

Property: The underline width.

Returns:
The current value. Default is `null`.

setFolderMouseOverUnderline

`public void setFolderMouseOverUnderline(Integer b)`

Property: The underline width.

Parameters:
`b` - The new value. Default is `null`.

getFolderSelectedBackground

`public java.awt.Paint getFolderSelectedBackground()`

Property: The background if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setFolderSelectedBackground

`public void setFolderSelectedBackground(java.awt.Paint p)`

(continued from last page)

Property: The background if the node label is selected.

Parameters:
`p` - The new value. Default is `null` which means same as when not selected.

getFolderSelectedFont

`public java.awt.Font getFolderSelectedFont()`

Property: The font if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setFolderSelectedFont

`public void setFolderSelectedFont(java.awt.Font f)`

Property: The font if the node label is selected.

Parameters:
`f` - The new value. Default is `null` which means same as when not selected.

getFolderSelectedForeground

`public java.awt.Color getFolderSelectedForeground()`

Property: The foreground if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setFolderSelectedForeground

`public void setFolderSelectedForeground(java.awt.Color c)`

Property: The foreground if the node label is selected.

Parameters:
`c` - The new value. Default is `null` which means same as when not selected.

getFolderSelectedUnderline

`public Integer getFolderSelectedUnderline()`

Property: The underline width if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setFolderSelectedUnderline

`public void setFolderSelectedUnderline(Integer width)`

Property: The underline width if the node label is selected.

Parameters:
`width` - The new value. Default is `null` which means same as when not selected.

getLeafUnderline

```
public Integer getLeafUnderline()
```

Property: The underline width in pixels.

Returns:
The current value. Default is `null` which means no underline.

setLeafUnderline

```
public void setLeafUnderline(Integer b)
```

Property: The underline width in pixels.

Parameters:
`b` - The current value. Default is `null` which means no underline.

getFolderUnderline

```
public Integer getFolderUnderline()
```

Property: The underline width in pixels.

Returns:
The current value. Default is `null` which means no underline.

setFolderUnderline

```
public void setFolderUnderline(Integer b)
```

Property: The underline width in pixels.

Parameters:
`b` - The current value. Default is `null` which means no underline.

getLeafBackground

```
public java.awt.Paint getLeafBackground()
```

Property: The background paint of the node.

Returns:
The current value. Default is `null` which means no background.

setLeafBackground

```
public void setLeafBackground(java.awt.Paint p)
```

Property: The background paint of the node.

Parameters:
`p` - The new value. Default is `null` which means no background.

getLeafFont

```
public java.awt.Font getLeafFont()
```

(continued from last page)

Property: The font of the label.

Returns:
The current value. Default is same as for the bean.

setLeafFont

```
public void setLeafFont(java.awt.Font f)
```

Property: The font of the label.

Parameters:
`f` - The new value. `null` reset it to the same as for the bean.

getLeafForeground

```
public java.awt.Color getLeafForeground()
```

Property: The foreground of the label.

Returns:
The current value. Default is same as for the bean.

setLeafForeground

```
public void setLeafForeground(java.awt.Color c)
```

Property: The foreground of the label.

Parameters:
`c` - The new value. `null` reset it to the same as for the bean.

getLeafCheckCursor

```
public java.awt.Cursor getLeafCheckCursor()
```

Property: The cursor set when the mouse is hovering over the check.

Returns:
The current value. Default is `null`.

setLeafCheckCursor

```
public void setLeafCheckCursor(java.awt.Cursor cur)
```

Property: The cursor set when the mouse is hovering over the check.

Parameters:
`cur` - The new value. Default is `null`.

getLeafCheckHalfSelectedIcon

```
public XtdImage getLeafCheckHalfSelectedIcon()
```

Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.

Returns:
The current value. Default is `HALF_CHECK_IMG` (private)

setLeafCheckHalfSelectedIcon

`public void setLeafCheckHalfSelectedIcon(XtdImage image)`

Property: The icon for when a check is "half" selected, which menas that some children are checked and some are not.

Parameters:
`image` - The new value. Default is `HALF_CHECK_IMG` (private)

getLeafCheckMouseOverBackground

`public java.awt.Paint getLeafCheckMouseOverBackground()`

Property: The background for when the mouse is hovering over the check.

Returns:
The current value. Default is `null`.

setLeafCheckMouseOverBackground

`public void setLeafCheckMouseOverBackground(java.awt.Paint p)`

Property: The background for when the mouse is hovering over the check.

Parameters:
`p` - The new value. Default is `null`.

getLeafCheckMouseOverForeground

`public java.awt.Paint getLeafCheckMouseOverForeground()`

Property: The foreground for when the mouse is hovering over the check.

Returns:
The current value. Default is `null`.

setLeafCheckMouseOverForeground

`public void setLeafCheckMouseOverForeground(java.awt.Paint p)`

Property: The foreground for when the mouse is hovering over the check.

Parameters:
`p` - The new value. Default is `null`.

isLeafCheckSelectable

`public boolean isLeafCheckSelectable()`

Property: If the check should be selectable.

Returns:
The current value. Default is `true`.

setLeafCheckSelectable

`public void setLeafCheckSelectable(boolean b)`

(continued from last page)

Property: If the check should be selectable.

Parameters:
`b` - The new value. Default is `true`.

getLeafCheckSelectedIcon

`public XtdImage getLeafCheckSelectedIcon()`

Property: The icon/image for when the check is selected.

Returns:
The current value. Default is `CHECK_IMG` (private)

setLeafCheckSelectedIcon

`public void setLeafCheckSelectedIcon(XtdImage image)`

Property: The icon/image for when the check is selected.

Parameters:
`image` - The new value. Default is `CHECK_IMG` (private)

getLeafLabelCursor

`public java.awt.Cursor getLeafLabelCursor()`

Property: The cursor for when the mouse is hovering over the label.

Returns:
The current value. Default is `null`.

setLeafLabelCursor

`public void setLeafLabelCursor(java.awt.Cursor cur)`

Property: The cursor for when the mouse is hovering over the label.

Parameters:
`cur` - The new value. Default is `null`.

isLeafLabelSelectable

`public boolean isLeafLabelSelectable()`

Property: If the label should be selectable.

Returns:
The current value. Default is `false`.

setLeafLabelSelectable

`public void setLeafLabelSelectable(boolean b)`

Property: If the label should be selectable.

Parameters:
`b` - The new value. Default is `false`.

getLeafMouseOverBackground

`public java.awt.Paint getLeafMouseOverBackground()`

Property: The background for when the mouse is hovering over the label.

Returns:
The current value. Default is `null`.

setLeafMouseOverBackground

`public void setLeafMouseOverBackground(java.awt.Paint p)`

Property: The background for when the mouse is hovering over the label.

Parameters:
`p` - The new value. Default is `null`.

getLeafMouseOverFont

`public java.awt.Font getLeafMouseOverFont()`

Property: The font for when the mouse is hovering over the label.

Returns:
The current value. Default is `null`.

setLeafMouseOverFont

`public void setLeafMouseOverFont(java.awt.Font f)`

Property: The font for when the mouse is hovering over the label.

Parameters:
`f` - The new value. Default is `null`.

getLeafMouseOverForeground

`public java.awt.Color getLeafMouseOverForeground()`

Property: The foreground for when the mouse is hovering over the label.

Returns:
The current value. Default is `null`.

setLeafMouseOverForeground

`public void setLeafMouseOverForeground(java.awt.Color c)`

Property: The foreground for when the mouse is hovering over the label.

Parameters:
`c` - The new value. Default is `null`.

getLeafMouseOverUnderline

`public Integer getLeafMouseOverUnderline()`

(continued from last page)

Property: The underline in pixels for when the mouse is hovering over the label.

Returns:
The current value. Default is `null`.

setLeafMouseOverUnderline

`public void setLeafMouseOverUnderline(Integer b)`

Property: The underline in pixels for when the mouse is hovering over the label.

Parameters:
`b` - The new value. Default is `null`.

getLeafSelectedBackground

`public java.awt.Paint getLeafSelectedBackground()`

Property: The background if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setLeafSelectedBackground

`public void setLeafSelectedBackground(java.awt.Paint p)`

Property: The background if the node label is selected.

Parameters:
`p` - The new value. Default is `null` which means same as when not selected.

getLeafSelectedFont

`public java.awt.Font getLeafSelectedFont()`

Property: The font if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setLeafSelectedFont

`public void setLeafSelectedFont(java.awt.Font f)`

Property: The font if the node label is selected.

Parameters:
`f` - The new value. Default is `null` which means same as when not selected.

getLeafSelectedForeground

`public java.awt.Color getLeafSelectedForeground()`

Property: The foreground if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setLeafSelectedForeground

```
public void setLeafSelectedForeground(java.awt.Color c)
```

Property: The foreground if the node label is selected.

Parameters:
`c` - The new value. Default is `null` which means same as when not selected.

getLeafSelectedUnderline

```
public Integer getLeafSelectedUnderline()
```

Property: The underline in pixels if the node label is selected.

Returns:
The current value. Default is `null` which means same as when not selected.

setLeafSelectedUnderline

```
public void setLeafSelectedUnderline(Integer width)
```

Property: The underline in pixels if the node label is selected.

Parameters:
`width` - The new value. Default is `null` which means same as when not selected.

isRootVisible

```
public boolean isRootVisible()
```

Property: If the root node should be visible.

Returns:
The old value. Default is `false`.

setRootVisible

```
public void setRootVisible(boolean b)
```

Property: If the root node should be visible.

Parameters:
`b` - The new value. Default is `false`.

revalidateNodes

```
public void revalidateNodes(Object fromCatId,
          boolean keepExpanded)
```

Revalidate the nodes starting from `fromCatId`.

Parameters:
`fromCatId` - The category id to start from. Not the Node or Category, but the ID.
`keepExpanded` - If the tree shold have the same expanded state after the call.

(continued from last page)

getRootCategoryId

```
public Object getRootCategoryId()
```

Returns the category ID of the root node.

Returns:

The category ID of the root node. `null` if the root node is the real root in the category depository.

---

setRootCategoryId

```
public void setRootCategoryId(Object catIDRoot)
```

Sets the category ID of the root node.

Parameters:

`catIDRoot` - The category ID of the root node. `null` if the root node should be the real root in the category depository.

---

getCheckedState

```
public static Boolean getCheckedState(PropertyKey key,
          Category cat)
```

Returns the checked state of a category. `null` means determinate because the category has sub categories where some are check and some are not checked. `Boolean.TRUE` means that this and all sub categories are selected. `Boolean.FALSE` means that this and all sub categories are un-selected.

Categories that has sub-categories does not maintain checked state. Their state are dependandt on the state of all sub-category leaves.

Parameters:

`key` - The key that is used to get the checked state from the Category. See `getCheckSelectedKey()`

`cat` - The category to check the state for.

Returns:

The state of the category and sub categories.

---

setCheckedState

```
public static void setCheckedState(PropertyKey key,
          Category cat,
          boolean b)
```

Set the selected state for the category. If a filder the state will propagate to all children.

Parameters:

`key` - The key that is used to get the checked state from the Category. See `getCheckSelectedKey()`

`cat` - The cageory to set the checked state on.

`b` - The new state of the check.

---

toggleCheckedState

```
public static void toggleCheckedState(PropertyKey key,
          Category cat)
```

Toggles the checked state on the category. If the category is a folder the folder and its childern will be set to "selected" if the folder itself or one or its childen are "unselected". Otherwise the category tree will be "unselected".

Parameters:

`key` - The key that is used to get the checked state from the Category. See `getCheckSelectedKey()`

(continued from last page)

`cat` - The category to toggle the selected state on.

interactionOccured

```
public void interactionOccured(InteractionEvent e)
```

addInteractionListener

```
public void addInteractionListener(InteractionListener l)
```

Adds a listener that listens to `InteractionEvents`. Interaction events are normally fired by the Interaction/Interactor/AbstractInteractionBroker framework, used for instance by the `com.miginfocom.ashape.shapes.AShape` framework.

The interaction events that is fired is when the user presses a check or the label in the tree.

Parameters:

`l` - The listener to add

See Also:

`CHECK_CLICKED`
`LABEL_CLICKED`

addInteractionListener

```
public void addInteractionListener(InteractionListener l,
          boolean asWeakRef)
```

Adds a listener that listens to `InteractionEvents`. Interaction events are normally fired by the Interaction/Interactor/AbstractInteractionBroker framework, used for instance by the `com.miginfocom.ashape.shapes.AShape` framework.

The interaction events that is fired is when the user presses a check or the label in the tree.

Parameters:

`l` - The listener to add

`asWeakRef` - If the listener should be added wrapped in a `java.lang.ref.WeakReference`. This defers memory leak problems since the garbage collector can collect the listener if it is only referenced from this list.

Note! This (weak reference) can not be used with listeners that doesn't have another real (a.k.a Strong) reference to it, as for instance an annonymous inner class. If one such listener is added it will be removed almost immediately by the garbage collector.

See Also:

`CHECK_CLICKED`
`LABEL_CLICKED`

removeInteractionListener

```
public void removeInteractionListener(InteractionListener l)
```

Removes the listener if it is added.

Parameters:

`l` - The listener to remove. If `null` nothing happens.

(continued from last page)

isIgnoreInteractionEvents

`public boolean isIgnoreInteractionEvents()`

Returns if events are currently ignored.

Returns:
If events are currently ignored.

See Also:
`setIgnoreInteractionEvents(boolean)`

---

setIgnoreInteractionEvents

`public boolean setIgnoreInteractionEvents(boolean b)`

Sets if events should be ignored, and thus not fired.

Parameters:
`b` - `true` turns off events

Returns:
The old state of this flag.

---

getDecorators

`public List getDecorators()`

Returns a cloned list with the decorators currently installed in this tree.

Returns:
The decorators. Not `null`.

---

setDecorators

`public void setDecorators(Collection decorators)`

Sets the decorators to use.

Parameters:
`decorators` - The decorators. Not `null`. List is cloned shallow for storage.

See Also:
`setDecorators(java.util.Collection)`

---

addDecorators

`public void addDecorators(Collection decorators)`

Calls `addDecorator(com.miginfocom.calendar.decorators.Decorator)` for every element in the collection.

Parameters:
`decorators` - The decorators. Not `null`.

See Also:
`addDecorators(java.util.Collection)`

(continued from last page)

sortDecorators

```
public void sortDecorators()
```

Resort the decorators that this tree handles. This is normally only needed if any of the decorators has changed its layer outside the control of this tree.

---

addDecorator

```
public void addDecorator(Decorator decorator)
```

Adds a decorator. A manual repaint has to be issued to paint the decorator.

Parameters:

`decorator` - The decorator. Insertion order is maintained and decorations will be painted in this order.

See Also:

`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

removeDecorator

```
public void removeDecorator(Decorator decorator)
```

Removes a decorator. A manual repaint has to be issued to paint the decorator.

Parameters:

`decorator` - The decorator to be removed.

See Also:

`removeDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

removeDecorator

```
public boolean removeDecorator(Class type,
          boolean inclSubClasses)
```

Removes the first decorator found with the class type `type`, including sub types id `inclSubClasses == true`.

Parameters:

`type` - The type. Not `null`.

`inclSubClasses` - If sub classes of `type` should be removed as well.

Returns:

If a decorator was removed.

---

removeDecorators

```
public void removeDecorators()
```

Removes all decorators. Note that this also removes the self decorating decorators. You should probably add `com.miginfocom.calendar.datearea.DefaultDateArea.MultiSelectRectangleDecorator` and `com.miginfocom.calendar.datearea.DefaultDateArea.ActivityViewDecorator` again. Add an instance by: `dateArea.addDecorator(dateArea.new ActivityViewDecorator(110));`

See Also:

`removeDecorators(java.util.Collection)`

(continued from last page)

removeDecorators

```
public void removeDecorators(Collection decorators)
```

Removes all decorators

Parameters:

`decorators` - The decorators to remove.

com.miginfocom.beans
Class DateAreaBean

```
java.lang.Object
   |
   +-DateAreaContainer
      |
      +-com.miginfocom.beans.DateAreaBean
```

public class **DateAreaBean**
extends DateAreaContainer

An object that extends `com.miginfocom.calendar.datearea.DateAreaContainer` to provide some easy-to-use bean properties to accomodate for "VB-like" visual programming.

It will cover a big chunk of all use cases but not all. It is possible to use this bean's properties to configure some parts and then use the `getDefaultDateArea()` to manually configure more advanced properties.

Constructor Summary

public	DateAreaBean()

Method Summary

void	addActivityDragResizeListener(ActivityDragResizeListener l) This call is transmitted to the addActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener)
void	addActivityDragResizeListener(ActivityDragResizeListener l, boolean asWeakRef) This call is transmitted to the addActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener, boolean)
void	addActivityMoveListener(ActivityMoveListener l) This call is transmitted to the addActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener)
void	addActivityMoveListener(ActivityMoveListener l, boolean asWeakRef) This call is transmitted to the addActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener, boolean)
void	addDateChangeListener(DateChangeListener l) This call is transmitted to the addDateChangeListener(com.miginfocom.util.dates.DateChangeListener)
void	addDateChangeListener(DateChangeListener l, boolean asWeakRef) This call is transmitted to the addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)
void	addInteractionListener(InteractionListener l) This call is transmitted to the addInteractionListener(com.miginfocom.ashape.interaction.InteractionListener)

void	addInteractionListener(InteractionListener l, boolean asWeakRef) This call is transmitted to the addInteractionListener(com.miginfocom.ashape.interaction.InteractionListener, boolean)
void	addNotify()
GridDimensionLayout	createLayout(Grid grid, int dimIx)
GridLineSpecification	createSpecification(DateArea dateArea)
void	doLayout()
String	getActivityDepositoryContext() Property: The context used as a key in the com.miginfocom.calendar.activity.ActivityDepository to get the activities that this date area should show.
ActivityLayout[]	getActivityLayouts() Returns the currently installed com.miginfocom.calendar.layout.ActivityLayouts.
String	getActivityPaintContext() Property: The default paint context that will be used to get an AShape to paint with.
java.awt.Paint	getBackgroundPaint() Property: The background paint in the date area.
boolean	getCategoryAutoRevalidate() Property: If the categories in the CategoryDepository changes the date area will revalidate itself if there is a category root set.
CategoryHeaderBean	getCategoryHeader() Returns the one and only category header bean.
Category	getCategoryRoot() Property: The category from the com.miginfocom.calendar.category.CategoryDepository that will serve as the root for this category header.
Object[]	getCategoryRootIDs() Property: The category IDs from the com.miginfocom.calendar.category.CategoryDepository that will serve as the root(s) for this category header.
boolean	getCategoryShowRoot() Property: If the root category set with setCategoryRoot(com.miginfocom.calendar.category.Category) should be shown or if its children should be shown in the first level.
javax.swing.border.Border	getDateAreaInnerBorder() Property: The inner border around the date area.
javax.swing.border.Border	getDateAreaOuterBorder() Property: The outer border around the date area.
DefaultDateArea	getDefaultDateArea()

DemoDataBean	getDemoDataBean() Property: A reference to a demo data bean that creates demo data for this date area.
java.awt.Paint	getDividerPaint() Property: The paint used to draw separator lines between some date ranges, for instance months.
int	getDividerRangeType() Property: The type of the divider range (e.g. day, week, month).
DateHeaderBean	getEastDateHeader() Returns the date header that is to the right the date area.
java.awt.Paint	getEvenBoundaryPaint() Property: The paint used to fill all even (0, 2, 4...) date ranges of the date area.
int	getEvenRangeType() Property: The date range type (i.e.
int	getEvenRangeTypeCount() Property: How many of getEvenRangeType() that should be bunched up and be called as one.
GridLineException[]	getHorizontalGridLineExceptions() Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
java.awt.Paint	getHorizontalGridLinePaintEven() Property: The paint used to draw the even (0, 2, 4...) horizontal grid lines.
java.awt.Paint	getHorizontalGridLinePaintOdd() Property: The paint used to draw the odd (1, 3, 5...) horizontal grid lines.
XtdImage	getImage() Property: The background or foreground image in the date area.
PlaceRect	getImagePlaceRect() Property: How the image should be placed relative to the bounds of the date area.
boolean	getImageTiling() Property: If the image should be tiled within the bounds.
AtRefRangeNumber	getLabelAlignX() Property: The horizontal alignment the label should have within the bounds denoted by the PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)).
AtRefRangeNumber	getLabelAlignY() Property: The vertical alignment the label should have within the bounds denoted by the PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)).
int	getLabelAntiAlias() Property: The anti aliasing hint used when drawing the label.
java.awt.Paint	getLabelBackground() Property: The java.awt.Paint used to draw the background of the label.
javax.swing.border.Border	getLabelBorder() Property: The border that is painted around the label.

int	getLabelCellModulo() Property: This property make is possible to show every second or third (for instance) cell label instead if the default every one.
String	getLabelDateFormat() Property: The date format that specifies the text that will be the label.
String	getLabelFirstDateFormat() Property: Date format for the "first" of something, e.g. first day of the month.
int	getLabelFirstInField() Property: If `getFirstDateFormat()` is !
java.awt.Font	getLabelFont() Property: The font used to draw the label.
java.awt.Color	getLabelForeground() Property: The paint that is used to draw the label.
java.awt.Dimension	getLabelMinimumCellSize() Property: If the cell is smaller than this size, in any dimension, the label will not be drawn.
java.awt.Paint	getLabelNowBackground() Property: The `java.awt.Paint` used to draw the background of the label for the cell that spans the current time.
javax.swing.border.Border	getLabelNowBorder() Property: The border that is painted around the label for the cell that spans the current time.
String	getLabelNowDateFormat() Property: The date format that specifies the text that will be the label for the cell that spans the current time.
java.awt.Font	getLabelNowFont() Property: The font used to draw the label for the cell that spans the current time.
java.awt.Color	getLabelNowForeground() Property: The paint that is used to draw the label for the cell that spans the current time.
Integer	getLabelNowRangeType() Property: The range type to round the "now" date to.
PlaceRect	getLabelPlaceRect() Property: Denotes the bounds that the label should get.
int	getLayerForActivities() Property: The layer index for the decorator that paints the activities.
int	getLayerForDividers() Property: The layer index for the decorator that paints the dividers between some date ranges.
int	getLayerForEvenFieldFill() Property: The layer index for the decorator that paints the even date ranges.
int	getLayerForGridLines() Property: The layer index for the decorator that paints the grid lines.

int	getLayerForImage() Property: The layer index for the decorator that paints a background/foreground image in the date area.
int	getLayerForLabels() Property: The layer index for the decorator that paints the labels in the cells.
int	getLayerForOccupied() Property: The layer index for the decorator that decorates the parts of the cells that has an activity which date range spans that time.
int	getLayerForOddFill() Property: The layer index for the decorator that paints the odd rows and/or columns.
int	getLayerForSelections() Property: The layer index for the decorator that paints the mouse over, selected and temporary pressed selections in the grid.
boolean	getMouseOverActivitiesOntop() Property: If Activites that have the mouse hovering over them should be painted on top of other activities.
boolean	getNoExpandedFolderGridLine() Property: If the grid lines around a folded (collapsed) folder sub row should be merged to one.
DateHeaderBean	getNorthDateHeader() Returns the date header that is above the date area.
java.awt.Paint	getOccupiedBackgroundPaint() Property: The background paint for the decoration where there's an activity which range overlaps.
boolean	getOccupiedMergeOverlapping() Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.
java.awt.Paint	getOccupiedNotBackgroundPaint() Property: The background paint for the decoration where there's an activity which range DOESN'T overlap.
java.awt.Paint	getOccupiedNotOutlinePaint() Property: The outline paint for the decoration where there's an activity which range DOESN'T overlap.
java.awt.Paint	getOccupiedOutlinePaint() Property: The outline paint for the decoration where there's an activity which range overlap.
PlaceRect	getOccupiedPlaceRect() Property: The placing of the rectangle that denoted the occupied/not occupied periods.
Integer	getOccupiedRoundToRangeType() Property: If occupied ranges should be rounded to a range type (e.g. a day or hour).
java.awt.Paint	getOddColumnPaint() Property: The paint to paint for all odd columns.
java.awt.Paint	getOddRowPaint() Property: The paint to paint for all odd rows.

int	getPrimaryDimension() What dimension time progresses if there was no wrapping.
int	getPrimaryDimensionCellType() Returns the type of `primDimCellSpanCount`.
int	getPrimaryDimensionCellTypeCount() Returns how may cells in the primary dimension.
GridLayoutProvider	getPrimaryDimensionLayout() Property: The layout provider for the primary dimension for the grid.
GridLayoutProvider	getSecondaryDimensionLayout() Property: The layout provider for the secondary dimension for the grid.
java.awt.Paint	getSelectablePaint() Property: The paint used to fill the cells that are marked as selectable.
boolean	getSelectedActivitiesOntop() Property: If Activites that are selected should be painted on top of other activities.
int	getSelectionBoundaryType() Property: The type of date range (e.g. day, hour, week) that selectionw will be rounded to.
java.awt.Paint	getSelectionMouseOverPaint() Property: The paint used to fill the cells that the mouse is over.
java.awt.Paint	getSelectionMousePressedPaint() Property: The paint used to fill the cells that the is pressed over (e.g. temporary selection).
java.awt.Paint	getSelectionPaint() Property: The paint used to fill the cells that are selected.
int	getSelectionType() Property: The type of selection that can be done on the date area background.
long	getSnapToMillis() Return what even millisecond to snap to when resizing and moving activities.
DateHeaderBean	getSouthDateHeader() Returns the date header that is below the date area.
SubRowCreator	getSubRowCreator() Property: A custom `com.miginfocom.calendar.grid.SubRowCreator` that can be set to create all sub rows for the date area bean.
java.awt.Paint	getSubRowGridLinePaint() Property: The paint used to draw sub row divider grid lines.
GridLineException[]	getVerticalGridLineExceptions() Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
java.awt.Paint	getVerticalGridLinePaintEven() Property: The paint used to draw the even (0, 2, 4...) vertical grid lines.
java.awt.Paint	getVerticalGridLinePaintOdd() Property: The paint used to draw the odd (1, 3, 5...) vertical grid lines.

DateHeaderBean	getWestDateHeader() Returns the date header that is to the left the date area.
String	getVisibleDateRangeString() Property: A string representing the visual date range of the exact form: "yyyyMMdd'T'HHmmssSSS'-'yyyyMMdd'T'HHmmssSSS" without the ' of course.
Integer	getWrapBoundary() Returns a which boundary (e.g. week, day) that the grid should wrap.
int	getWrapBoundaryCount() Returns the number of wrapBoundary before wrapping.
boolean	isDesignTimeHelp() Property: If true draws an information message in "design time".
boolean	isShowNoFitIcon() Property: If the little icon that shows that not all activities have fitted within a date range (normally a day).
void	paint(java.awt.Graphics g)
void	removeActivityDragResizeListener(ActivityDragResizeListener l) This call is transmitted to the removeActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener)
void	removeActivityMoveListener(ActivityMoveListener l) This call is transmitted to the removeActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener)
void	removeDateChangeListener(DateChangeListener l) This call is transmitted to the removeDateChangeListener(com.miginfocom.util.dates.DateChangeListener)
void	removeInteractionListener(InteractionListener l) This call is transmitted to the removeInteractionListener(com.miginfocom.ashape.interaction.InteractionListener).
void	removeNotify()
void	revalidateGrid() Recreates the grid in the date area
void	setActivityDepositoryContext(String context) Property: The context used as a key in the com.miginfocom.calendar.activity.ActivityDepository to get the activities that this date area should show.
void	setActivityLayouts(ActivityLayout[] layouts) Sets the com.miginfocom.calendar.layout.ActivityLayouts.
void	setActivityPaintContext(String ctx) Property: The default paint context that will be used to get an AShape to paint with.
void	setBackgroundPaint(java.awt.Paint p) Property: The background paint in the date area.

void	setCategoryAutoRevalidate(boolean b) Property: If the categories in the CategoryDepository changes the date area will revalidate itself if there is a category root set.
void	setCategoryHeader(CategoryHeaderBean header) Sets or disables the category header.
void	setCategoryRoot(Category root) Property: The category from the com.miginfocom.calendar.category.CategoryDepository that will serve as the root for this category header.
void	setCategoryRootIDs(Object[] rootIDs) Property: The category IDs from the com.miginfocom.calendar.category.CategoryDepository that will serve as the root(s) for this category header.
void	setCategoryShowRoot(boolean b) Property: If the root category set with setCategoryRoot(com.miginfocom.calendar.category.Category) should be shown or if its children should be shown in the first level.
void	setDateAreaInnerBorder(javax.swing.border.Border b) Property: The inner border around the date area.
void	setDateAreaOuterBorder(javax.swing.border.Border b) Property: The outer border around the date area.
void	setDemoDataBean(DemoDataBean b) Property: A reference to a demo data bean that creates demo data for this date area.
void	setDesignTimeHelp(boolean b) Property: If true draws an information message in "design time".
void	setDividerPaint(java.awt.Paint paint) Property: The paint used to draw separator lines between some date ranges, for instance months.
void	setDividerRangeType(int rangeType) Property: The type of the divider range (e.g. day, week, month).
void	setEastDateHeader(DateHeaderBean header) Sets the date header that will be to the right the date area.
void	setEvenBoundaryPaint(java.awt.Paint paint) Property: The paint used to fill all even (0, 2, 4...) date ranges of the date area.
void	setEvenRangeType(int rangeType) Property: The date range type (i.e.
void	setEvenRangeTypeCount(int count) Property: How many of getEvenRangeType() that should be bunched up and be called as one.
void	setHorizontalGridLineExceptions(GridLineException[] exceptions) Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
void	setHorizontalGridLinePaintEven(java.awt.Paint paint) Property: The paint used to draw the even (0, 2, 4...) horizontal grid lines.

void	setHorizontalGridLinePaintOdd(java.awt.Paint paint) Property: The paint used to draw the odd (1, 3, 5...) horizontal grid lines.
void	setImage(XtdImage img) Property: The background or foreground image in the date area.
void	setImagePlaceRect(PlaceRect r) Property: How the image should be placed relative to the bounds of the date area.
void	setImageTiling(boolean b) Property: If the image should be tiled within the bounds.
void	setLabelAlignX(AtRefRangeNumber x) Property: The horizontal alignment the label should have within the bounds denoted by the PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)).
void	setLabelAlignY(AtRefRangeNumber y) Property: The vertical alignment the label should have within the bounds denoted by the PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)).
void	setLabelAntiAlias(int hint) Property: The anti aliasing hint used when drawing the label.
void	setLabelBackground(java.awt.Paint paint) Property: The java.awt.Paint used to draw the background of the label.
void	setLabelBorder(javax.swing.border.Border b) Property: The border that is painted around the label.
void	setLabelCellModulo(int i) Property: This property make is possible to show every second or third (for instance) cell label instead if the default every one.
void	setLabelDateFormat(String format) Property: The date format that specifies the text that will be the label.
void	setLabelFirstDateFormat(String format) Property: Date format for the "first" of something, e.g. first day of the month.
void	setLabelFirstInField(int calField) Property: If getFirstDateFormat() is !
void	setLabelFont(java.awt.Font font) Property: The font used to draw the label.
void	setLabelForeground(java.awt.Color c) Property: The paint that is used to draw the label.
void	setLabelMinimumCellSize(java.awt.Dimension size) Property: If the cell is smaller than this size, in any dimension, the label will not be drawn.
void	setLabelNowBackground(java.awt.Paint paint) Property: The java.awt.Paint used to draw the background of the label for the cell that spans the current time.
void	setLabelNowBorder(javax.swing.border.Border b) Property: The border that is painted around the label for the cell that spans the current time.

void	setLabelNowDateFormat(String format) Property: The date format that specifies the text that will be the label for the cell that spans the current time.
void	setLabelNowFont(java.awt.Font font) Property: The font used to draw the label for the cell that spans the current time.
void	setLabelNowForeground(java.awt.Color c) Property: The paint that is used to draw the label for the cell that spans the current time.
void	setLabelNowRangeType(Integer rangeType) Property: The range type to round the "now" date to.
void	setLabelPlaceRect(PlaceRect placeRect) Property: Denotes the bounds that the label should get.
void	setLayerForActivities(int i) Property: The layer index for the decorator that paints the activities.
void	setLayerForDividers(int i) Property: The layer index for the decorator that paints the dividers between some date ranges.
void	setLayerForEvenFieldFill(int i) Property: The layer index for the decorator that paints the even date ranges.
void	setLayerForGridLines(int i) Property: The layer index for the decorator that paints the grid lines.
void	setLayerForImage(int i) Property: The layer index for the decorator that paints a background/foreground image in the date area.
void	setLayerForLabels(int i) Property: The layer index for the decorator that paints the labels in the cells.
void	setLayerForOccupied(int i) Property: The layer index for the decorator that decorates the parts of the cells that has an activity which date range spans that time.
void	setLayerForOddFill(int i) Property: The layer index for the decorator that paints the odd rows and/or columns.
void	setLayerForSelections(int i) Property: The layer index for the decorator that paints the mouse over, selected and temporary pressed selections in the grid.
void	setMouseOverActivitiesOntop(boolean b) Property: If Activites that have the mouse hovering over them should be painted on top of other activities.
void	setNoExpandedFolderGridLine(boolean b) Property: If the grid lines around a folded (collapsed) folder sub row should be merged to one.
void	setNorthDateHeader(DateHeaderBean header) Sets the date header that will be above the date area.
void	setOccupiedBackgroundPaint(java.awt.Paint p) Property: The background paint for the decoration where there's an activity which range overlap.

void	setOccupiedMergeOverlapping(boolean b) Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.
void	setOccupiedNotBackgroundPaint(java.awt.Paint p) Property: The background paint for the decoration where there's an activity which range DOESN'T overlap.
void	setOccupiedNotOutlinePaint(java.awt.Paint p) Property: The outline paint for the decoration where there's an activity which range DOESN'T overlap.
void	setOccupiedOutlinePaint(java.awt.Paint p) Property: The outline paint for the decoration where there's an activity which range overlap.
void	setOccupiedPlaceRect(PlaceRect r) Property: The placing of the rectangle that denoted the occupied/not occupied periods.
void	setOccupiedRoundToRangeType(Integer rangeType) Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.
void	setOddColumnPaint(java.awt.Paint paint) Property: The paint to paint for all odd columns.
void	setOddRowPaint(java.awt.Paint paint) Property: The paint to paint for all odd rows.
void	setPrimaryDimension(int dim) Sets the dimension time progresses if there was no wrapping.
void	setPrimaryDimensionCellType(int cellType) Sets the type of primDimCellSpanCount.
void	setPrimaryDimensionCellTypeCount(int count) Sets how many primDimCellSpanType a cell spanns in the primary dimension.
void	setPrimaryDimensionLayout(GridLayoutProvider layout) Property: The layout provider for the primary dimension for the grid.
void	setSecondaryDimensionLayout(GridLayoutProvider layout) Property: The layout provider for the secondary dimension for the grid.
void	setSelectablePaint(java.awt.Paint paint) Property: The paint used to fill the cells that are marked as selectable.
void	setSelectedActivitiesOntop(boolean b) Property: If Activites that are selected should be painted on top of other activities.
void	setSelectionBoundaryType(int rangeType) Property: The type of date range (e.g. day, hour, week) that selectionw will be rounded to.
void	setSelectionMouseOverPaint(java.awt.Paint paint) Property: The paint used to fill the cells that the mouse is over.
void	setSelectionMousePressedPaint(java.awt.Paint paint) Property: The paint used to fill the cells that the is pressed over (e.g. temporary selection).
void	setSelectionPaint(java.awt.Paint paint) Property: The paint used to fill the cells that are selected.

void	setSelectionType(int type) Property: The type of selection that can be done on the date area background.
void	setShowNoFitIcon(boolean b) Property: If the little icon that shows that not all activities have fitted within a date range (normally a day).
void	setSnapToMillis(long millis) Sets what even millisecond to snap to when resizing and moving activities.
void	setSouthDateHeader(DateHeaderBean header) Sets the date header that will be below the date area.
void	setSubRowCreator(SubRowCreator creator) Property: A custom com.miginfocom.calendar.grid.SubRowCreator that can be set to create all sub rows for the date area bean.
void	setSubRowGridLinePaint(java.awt.Paint paint) Property: The paint used to draw sub row divider grid lines.
void	setVerticalGridLineExceptions(GridLineException[] exceptions) Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
void	setVerticalGridLinePaintEven(java.awt.Paint paint) Property: The paint used to draw the even (0, 2, 4...) vertical grid lines.
void	setVerticalGridLinePaintOdd(java.awt.Paint paint) Property: The paint used to draw the odd (1, 3, 5...) vertical grid lines.
void	setWestDateHeader(DateHeaderBean header) Sets the date header that will be to the left the date area.
void	setVisibleDateRangeString(String dateStrings) Property: A string representing the visual date range of the exact form: "yyyyMMdd'T'HHmmssSSS'-'yyyyMMdd'T'HHmmssSSS" without the ' of course.
void	setWrapBoundary(Integer wrap) Set which boundary (e.g. week, day) that the grid should wrap.
void	setWrapBoundaryCount(int count) Sets the number of wrapBoundary before wrapping.
void	validateHeaders()

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

DateAreaBean

public **DateAreaBean**()

(continued from last page)

Methods

addNotify

```
public void addNotify()
```

removeNotify

```
public void removeNotify()
```

createLayout

```
public GridDimensionLayout createLayout(Grid grid,
         int dimIx)
```

getDefaultDateArea

```
public DefaultDateArea getDefaultDateArea()
```

Returns:

The `DefaultDateArea`.

createSpecification

```
public GridLineSpecification createSpecification(DateArea dateArea)
```

paint

```
public void paint(java.awt.Graphics g)
```

doLayout

```
public void doLayout()
```

revalidateGrid

```
public void revalidateGrid()
```

Recreates the grid in the date area

isDesignTimeHelp

```
public boolean isDesignTimeHelp()
```

(continued from last page)

Property: If `true` draws an information message in "design time".

Returns:
The current value. `true` is default.

---

setDesignTimeHelp

`public void setDesignTimeHelp(boolean b)`

Property: If `true` draws an information message in "design time".

Parameters:
`b` - The new value. `true` is default.

---

getVisibleDateRangeString

`public String getVisibleDateRangeString()`

Property: A string representing the visual date range of the exact form: `"yyyyMMdd'T'HHmmssSSS'-'yyyyMMdd'T'HHmmssSSS"` without the ' of course. Example: `20061030T164500000`

Note that the dates will be rounded to even up to the cells defined by `primaryDimensionCellType` and `primaryDimensionCellTypeCount` as well as `wrapType`.

Note that this is a convenience that should only be used for tests since there is no time zone or locale information.

If there is a format error the property set will be silently ignored.

Returns:
The current visual date range. Not `null`.

See Also:
`setVisibleDateRange(com.miginfocom.util.dates.DateRangeI)`

---

setVisibleDateRangeString

`public void setVisibleDateRangeString(String dateStrings)`

Property: A string representing the visual date range of the exact form: `"yyyyMMdd'T'HHmmssSSS'-'yyyyMMdd'T'HHmmssSSS"` without the ' of course. Example: `20061030T164500000`

Note that this is a convenience that should only be used for tests since there is no time zone or locale information.

If there is a format error the property set will be silently ignored.

This property should be set **after** the structure if the date area (grid) is set with the properties that start with primaryXXX

Parameters:
`dateStrings` - The new visual date range. Not `null`.

See Also:
`getVisibleDateRange()`
`getVisibleDateRangeCorrected()`

---

getCategoryRoot

`public Category getCategoryRoot()`

(continued from last page)

Property: The category from the `com.miginfocom.calendar.category.CategoryDepository` that will serve as the root for this category header. Note that this category can either be shown itself or just serve as the invisible root to show its children as a multi root. This can be change with `setCategoryShowRoot(boolean)`.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Returns:

The current root category. May be `null` which means that categories is generally disabled for this bean.

---

setCategoryRoot

`public void` **`setCategoryRoot`**`(Category root)`

Property: The category from the `com.miginfocom.calendar.category.CategoryDepository` that will serve as the root for this category header. Note that this category can either be shown itself or just serve as the invisible root to show its children as a multi root. This can be change with `setCategoryShowRoot(boolean)`.

The bean `DemoDataBean` is a `Category` so if one of those is created to construct demo data it can be set here as the root.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Parameters:

`root` - The new root category. May be `null` which means that categories is generally disabled for this bean.

---

getCategoryRootIDs

`public Object[]` **`getCategoryRootIDs`**`()`

Property: The category IDs from the `com.miginfocom.calendar.category.CategoryDepository` that will serve as the root(s) for this category header. Note that this category(s) can either be shown itself or just serve as the invisible root(s) to show its children as a multi root. This can be change with `setCategoryShowRoot(boolean)`.

The bean `DemoDataBean` is a `Category` so if one of those is created to construct demo data it can be set here as the root.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Returns:

The current root category. May be `null` which means that categories is generally disabled for this bean.

---

setCategoryRootIDs

`public void` **`setCategoryRootIDs`**`(Object[] rootIDs)`

Property: The category IDs from the `com.miginfocom.calendar.category.CategoryDepository` that will serve as the root(s) for this category header. Note that this category(s) can either be shown itself or just serve as the invisible root(s) to show its children as a multi root. This can be change with `setCategoryShowRoot(boolean)`.

The bean `DemoDataBean` is a `Category` so if one of those is created to construct demo data it can be set here as the root.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Parameters:

`rootIDs` - The new category root ID(s). May be `null` which means that categories is generally disabled for this bean.

---

getCategoryAutoRevalidate

`public boolean` **`getCategoryAutoRevalidate`**`()`

Property: If the categories in the `CategoryDepository` changes the date area will revalidate itself if there is a category root set.

Returns:

If the auto revalidation is on.

setCategoryAutoRevalidate

`public void setCategoryAutoRevalidate(boolean b)`

Property: If the categories in the `CategoryDepository` changes the date area will revalidate itself if there is a category root set.

Parameters:

`b` - If the auto revalidation is on.

getSubRowCreator

`public SubRowCreator getSubRowCreator()`

Property: A custom `com.miginfocom.calendar.grid.SubRowCreator` that can be set to create all sub rows for the date area bean. Note that if this is set then the `categoryRoot`, `categoryRoots` and `categoryShowRoot` properties will not be used since they are only used for the default sub row creator (which is a CategorySubRowCreator). The default value for this property is `null` and then the two properties mentioned above will be used to create the sub rows.

Returns:

The current creator. Default is `null`.

See Also:

`setCategoryRoot(com.miginfocom.calendar.category.Category)`
`setCategoryShowRoot(boolean)`

setSubRowCreator

`public void setSubRowCreator(SubRowCreator creator)`

Property: A custom `com.miginfocom.calendar.grid.SubRowCreator` that can be set to create all sub rows for the date area bean. Note that if this is set then the `categoryRoot`, `categoryRoots` and `categoryShowRoot` properties will not be used since they are only used for the default sub row creator (which is a CategorySubRowCreator). The default value for this property is `null` and then the two properties mentioned above will be used to create the sub rows.

Parameters:

`creator` - The new creator. Default is `null`.

See Also:

`setCategoryRoot(com.miginfocom.calendar.category.Category)`
`setCategoryShowRoot(boolean)`

getCategoryShowRoot

`public boolean getCategoryShowRoot()`

Property: If the root category set with `setCategoryRoot(com.miginfocom.calendar.category.Category)` should be shown or if its children should be shown in the first level.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Returns:

If the root category should be show as the sole root.

setCategoryShowRoot

`public void setCategoryShowRoot(boolean b)`

(continued from last page)

Property: If the root category set with `setCategoryRoot(com.miginfocom.calendar.category.Category)` should be shown or if its children should be shown in the first level.

NOTE! Not used if `setSubRowCreator(com.miginfocom.calendar.grid.SubRowCreator)` is `!= null`!

Parameters:

`b` - If the root category should be show as the sole root.

---

getLayerForActivities

`public int` **`getLayerForActivities`**`()`

Property: The layer index for the decorator that paints the activities.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`DefaultDateArea.ActivityViewDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForActivities

`public void` **`setLayerForActivities`**`(int i)`

Property: The layer index for the decorator that paints the activities.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`DefaultDateArea.ActivityViewDecorator`

---

getLayerForGridLines

`public int` **`getLayerForGridLines`**`()`

Property: The layer index for the decorator that paints the grid lines.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`GridLineDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForGridLines

`public void` **`setLayerForGridLines`**`(int i)`

(continued from last page)

Property: The layer index for the decorator that paints the grid lines.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`GridLineDecorator`

getLayerForDividers

```
public int getLayerForDividers()
```

Property: The layer index for the decorator that paints the dividers between some date ranges. For example a darker line between months.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`DateSeparatorDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

setLayerForDividers

```
public void setLayerForDividers(int i)
```

Property: The layer index for the decorator that paints the dividers between some date ranges. For example a darker line between months.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`DateSeparatorDecorator`

getLayerForSelections

```
public int getLayerForSelections()
```

Property: The layer index for the decorator that paints the mouse over, selected and temporary pressed selections in the grid. It has nothing to do with selection of activities, but all to do with selection of cells in the date grid.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

(continued from last page)

See Also:
`SelectionGridDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForSelections

`public void setLayerForSelections(int i)`

Property: The layer index for the decorator that paints the mouse over, selected and temporary pressed selections in the grid. It has nothing to do with selection of activities, but all to do with selection of cells in the date grid.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:
`i` - The new layer for the decorator.

See Also:
`SelectionGridDecorator`

---

getLayerForLabels

`public int getLayerForLabels()`

Property: The layer index for the decorator that paints the labels in the cells. For instance the day number or time.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:
The current index.

See Also:
`CellLabelDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForLabels

`public void setLayerForLabels(int i)`

Property: The layer index for the decorator that paints the labels in the cells. For instance the day number or time.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:
`i` - The new layer for the decorator.

See Also:
`CellLabelDecorator`

---

getLayerForEvenFieldFill

`public int getLayerForEvenFieldFill()`

(continued from last page)

Property: The layer index for the decorator that paints the even date ranges. It can for instance be used to draw every other month in a darker color.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`EvenFieldFillDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForEvenFieldFill

```
public void setLayerForEvenFieldFill(int i)
```

Property: The layer index for the decorator that paints the even date ranges. It can for instance be used to draw every other month in a darker color.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`EvenFieldFillDecorator`

---

getLayerForOccupied

```
public int getLayerForOccupied()
```

Property: The layer index for the decorator that decorates the parts of the cells that has an activity which date range spans that time. For instance Microsoft's Outlook has this in it's day(s) view. It paints blue on the left side of those cells. This decoration is quite flexible and can paint the background for the whole cells or just parts of it.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`OccupiedDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

---

setLayerForOccupied

```
public void setLayerForOccupied(int i)
```

Property: The layer index for the decorator that decorates the parts of the cells that has an activity which date range spans that time. For instance Microsoft's Outlook has this in it's day(s) view. It paints blue on the left side of those cells. This decoration is quite flexible and can paint the background for the whole cells or just parts of it.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

(continued from last page)

Parameters:

`i` - The new layer for the decorator.

See Also:

`OccupiedDecorator`

getLayerForImage

`public int getLayerForImage()`

Property: The layer index for the decorator that paints a background/foreground image in the date area.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`ImageDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

setLayerForImage

`public void setLayerForImage(int i)`

Property: The layer index for the decorator that paints a background/foreground image in the date area.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`ImageDecorator`

getLayerForOddFill

`public int getLayerForOddFill()`

Property: The layer index for the decorator that paints the odd rows and/or columns. It can for instance be used to draw every other row in a darker color.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Returns:

The current index.

See Also:

`OddRowFillDecorator`
`addDecorator(com.miginfocom.calendar.decorators.Decorator)`

setLayerForOddFill

`public void setLayerForOddFill(int i)`

(continued from last page)

Property: The layer index for the decorator that paints the odd rows and/or columns. It can for instance be used to draw every other row in a darker color.

This property affect the order in which the different layers are painted, and thus what gets painted on top of what. Lower layer indexes will be painted first and will end up below layers with higher index. Everything that is painted in a date area is painted in layers, and the layers are implemented with `com.miginfocom.calendar.decorators.Decorators`.

Parameters:

`i` - The new layer for the decorator.

See Also:

`OddRowFillDecorator`

---

getDemoDataBean

`public DemoDataBean getDemoDataBean()`

Property: A reference to a demo data bean that creates demo data for this date area. This is normally only used for testing purposes and should not be used in any production environment.

Returns:

The current value of the property or `null`.

---

setDemoDataBean

`public void setDemoDataBean(DemoDataBean b)`

Property: A reference to a demo data bean that creates demo data for this date area. This is normally only used for testing purposes and should not be used in any production environment.

Parameters:

`b` - The new value for the property. May be `null`.

---

getImage

`public XtdImage getImage()`

Property: The background or foreground image in the date area. Note that `com.miginfocom.util.gfx.XtdImage` is very flexible and can be created in may ways, including from an enbedded BASE64 string.

This is a property that affects an optional background or foreground image in the date area.

Returns:

The current value of the property.

See Also:

`ImageDecorator`

---

setImage

`public void setImage(XtdImage img)`

Property: The background or foreground image in the date area. Note that `com.miginfocom.util.gfx.XtdImage` is very flexible and can be created in may ways, including from an enbedded BASE64 string.

This is a property that affects an optional background or foreground image in the date area.

Parameters:

`img` - The new value for the property. May be `null`.

See Also:

(continued from last page)

`ImageDecorator`

getImagePlaceRect

`public PlaceRect **getImagePlaceRect**()`

Property: How the image should be placed relative to the bounds of the date area. Note that the placing is very flexible through the use of the rectangle transformation interface `PlaceRect`.

This is a property that affects an optional background or foreground image in the date area.

Returns:
The current value of the property.

See Also:
`ImageDecorator`
`com.miginfocom.util.gfx.geometry.AbsRect`
`com.miginfocom.util.gfx.geometry.AlignRect`
`com.miginfocom.util.gfx.geometry.AspectRatioRect`

setImagePlaceRect

`public void **setImagePlaceRect**(PlaceRect r)`

Property: How the image should be placed relative to the bounds of the date area. Note that the placing is very flexible through the use of the rectangle transformation interface `PlaceRect`.

This is a property that affects an optional background or foreground image in the date area.

Parameters:
`r` - The new value for the property. May be `null`.

See Also:
`ImageDecorator`
`com.miginfocom.util.gfx.geometry.AbsRect`
`com.miginfocom.util.gfx.geometry.AlignRect`
`com.miginfocom.util.gfx.geometry.AspectRatioRect`

getImageTiling

`public boolean **getImageTiling**()`

Property: If the image should be tiled within the bounds. It is stretched/shrinked otherwise, depending on the rectangle returned bu the `placeRect`.

This is a property that affects an optional background or foreground image in the date area.

Returns:
The current value of the property.

See Also:
`ImageDecorator`

setImageTiling

`public void **setImageTiling**(boolean b)`

Property: If the image should be tiled within the bounds. It is stretched/shrinked otherwise, depending on the rectangle returned bu the `placeRect`.

This is a property that affects an optional background or foreground image in the date area.

(continued from last page)

Parameters:
b - The new value for the property.

See Also:
ImageDecorator

---

getOccupiedBackgroundPaint

```
public java.awt.Paint getOccupiedBackgroundPaint()
```

Property: The background paint for the decoration where there's an activity which range overlaps.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:
The current value of the property.

See Also:
OccupiedDecorator

---

setOccupiedBackgroundPaint

```
public void setOccupiedBackgroundPaint(java.awt.Paint p)
```

Property: The background paint for the decoration where there's an activity which range overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:
p - The new value for the property.

See Also:
OccupiedDecorator

---

getOccupiedOutlinePaint

```
public java.awt.Paint getOccupiedOutlinePaint()
```

Property: The outline paint for the decoration where there's an activity which range overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:
The current value of the property.

See Also:
OccupiedDecorator

---

setOccupiedOutlinePaint

```
public void setOccupiedOutlinePaint(java.awt.Paint p)
```

(continued from last page)

Property: The outline paint for the decoration where there's an activity which range overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:

`p` - The new value for the property.

See Also:

`OccupiedDecorator`

---

getOccupiedNotBackgroundPaint

```
public java.awt.Paint getOccupiedNotBackgroundPaint()
```

Property: The background paint for the decoration where there's an activity which range DOESN'T overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:

The current value of the property.

See Also:

`OccupiedDecorator`

---

setOccupiedNotBackgroundPaint

```
public void setOccupiedNotBackgroundPaint(java.awt.Paint p)
```

Property: The background paint for the decoration where there's an activity which range DOESN'T overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:

`p` - The new value for the property.

See Also:

`OccupiedDecorator`

---

getOccupiedNotOutlinePaint

```
public java.awt.Paint getOccupiedNotOutlinePaint()
```

Property: The outline paint for the decoration where there's an activity which range DOESN'T overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:

The current value of the property.

See Also:

`OccupiedDecorator`

(continued from last page)

setOccupiedNotOutlinePaint

```
public void setOccupiedNotOutlinePaint(java.awt.Paint p)
```

Property: The outline paint for the decoration where there's an activity which range DOESN'T overlap.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:

`p` - The new value for the property.

See Also:

`OccupiedDecorator`

---

getOccupiedPlaceRect

```
public PlaceRect getOccupiedPlaceRect()
```

Property: The placing of the rectangle that denoted the occupied/not occupied periods. The reference rectangle is normally the size of the activity range in the primary dimension and the whole cell in the secondary dimension.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:

The current value of the property.

See Also:

`OccupiedDecorator`

---

setOccupiedPlaceRect

```
public void setOccupiedPlaceRect(PlaceRect r)
```

Property: The placing of the rectangle that denoted the occupied/not occupied periods. The reference rectangle is normally the size of the activity range in the primary dimension and the whole cell in the secondary dimension.

E.g:

```
new AbsRect(SwingConstants.BOTTOM, new Integer(10)) or
 new AbsRect(SwingConstants.LEFT, new Integer(10))
```

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:

`r` - The new value for the property.

See Also:

`OccupiedDecorator`

---

getOccupiedMergeOverlapping

```
public boolean getOccupiedMergeOverlapping()
```

(continued from last page)

Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:
The current value of the property.

See Also:
`OccupiedDecorator`

---

setOccupiedMergeOverlapping

```
public void setOccupiedMergeOverlapping(boolean b)
```

Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:
`b` - The new value for the property.

See Also:
`OccupiedDecorator`

---

getOccupiedRoundToRangeType

```
public Integer getOccupiedRoundToRangeType()
```

Property: If occupied ranges should be rounded to a range type (e.g. a day or hour). This is for showing (for instance) that a day is "occupied" even if just a part of it is.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Returns:
The current value of the property. E.g. `RANGE_TYPE_DAY`. May be `null`.

See Also:
`OccupiedDecorator`

---

setOccupiedRoundToRangeType

```
public void setOccupiedRoundToRangeType(Integer rangeType)
```

Property: If occupied activity ranges that overlap should be combined (merged) before draing the occupied rects.

This is a property that affects the visual representation of the occupied decorator. It can be used to paint different backgrounds/outlines for parts is the date area that has a activity overlapping and not. For instance the blue left side in Microoft Outlook's day views can easily be simulated with this.

Parameters:
`rangeType` - The new value for the property. If `null` there will be no rounding (default).

See Also:
`OccupiedDecorator`

getActivityPaintContext

`public String getActivityPaintContext()`

Property: The default paint context that will be used to get an AShape to paint with. This is the default context to use (`null` is default) if it is not set on the activity directly with `setPaintContext(String)`.

It can for instance be used to "connect" this bean's visual appearence to that of a `ActivityAShapeBean`'s `paintContext`.

Returns:
The current value of the property.

See Also:
`getDefaultPaintContext()`
`ActivityAShapeBean.getPaintContext()`

Since:
6.0

setActivityPaintContext

`public void setActivityPaintContext(String ctx)`

Property: The default paint context that will be used to get an AShape to paint with. This is the default context to use (`null` is default) if it is not set on the activity directly with `setPaintContext(String)`.

It can for instance be used to "connect" this bean's visual appearence to that of a `ActivityAShapeBean`'s `paintContext`.

Parameters:
`ctx` - The new value for the property. May be `null`.

See Also:
`setDefaultPaintContext(String)`
`ActivityAShapeBean.setPaintContext(String)`

Since:
6.0

getPrimaryDimensionLayout

`public GridLayoutProvider getPrimaryDimensionLayout()`

Property: The layout provider for the primary dimension for the grid. The primary dimension is the one (vertical or horizontal) that would be the only one if there was no wrapping. For instance normal text has horizontal as primary dimension.

Note that normally a `GridLayoutProvider` can privide for both dimensions, however to be able to set different types of layouts in a visual environment this one will only be used to provide a layout for one dimension. This class (`DateAreaBean`) is itself a `GridLayoutProvider` and will relay the request to this provider, if set, but only for one dimension.

Returns:
The layout provider for the primary dimension.

setPrimaryDimensionLayout

`public void setPrimaryDimensionLayout(GridLayoutProvider layout)`

(continued from last page)

Property: The layout provider for the primary dimension for the grid. The primary dimension is the one (vertical or horizontal) that would be the only one if there was no wrapping. For instance normal text has horizontal as primary dimension.

Note that normally a `GridLayoutProvider` can privide for both dimensions, however to be able to set different types of layouts in a visual environment this one will only be used to provide a layout for one dimension. This class (`DateAreaBean`) is itself a `GridLayoutProvider` and will relay the request to this provider, if set, but only for one dimension.

Parameters:
`layout`

---

getSecondaryDimensionLayout

```
public GridLayoutProvider getSecondaryDimensionLayout()
```

Property: The layout provider for the secondary dimension for the grid. The secondary dimension is the one (vertical or horizontal) that increases when there is a wrap. For instance normal text has vertical as secondary dimension.

Note that normally a `GridLayoutProvider` can privide for both dimensions, however to be able to set different types of layouts in a visual environment this one will only be used to provide a layout for one dimension. This class (`DateAreaBean`) is itself a `GridLayoutProvider` and will relay the request to this provider, if set, but only for one dimension.

Returns:
The layout provider for the primary dimension.

---

setSecondaryDimensionLayout

```
public void setSecondaryDimensionLayout(GridLayoutProvider layout)
```

Property: The layout provider for the secondary dimension for the grid. The secondary dimension is the one (vertical or horizontal) that increases when there is a wrap. For instance normal text has vertical as secondary dimension.

Note that normally a `GridLayoutProvider` can privide for both dimensions, however to be able to set different types of layouts in a visual environment this one will only be used to provide a layout for one dimension. This class (`DateAreaBean`) is itself a `GridLayoutProvider` and will relay the request to this provider, if set, but only for one dimension.

Parameters:
`layout`

---

getWrapBoundary

```
public Integer getWrapBoundary()
```

Returns a which boundary (e.g. week, day) that the grid should wrap. E.g `RANGE_TYPE_WEEK`.

Returns:
Which boundary (e.g. week, day) that the grid should wrap. Default is `DateRangeI.RANGE_TYPE_WEEK`.

---

setWrapBoundary

```
public void setWrapBoundary(Integer wrap)
```

Set which boundary (e.g. week, day) that the grid should wrap. E.g `RANGE_TYPE_WEEK`.

Parameters:
`wrap` - Which boundary (e.g. week, day) that the grid should wrap. Default is `DateRangeI.RANGE_TYPE_WEEK`.

(continued from last page)

getWrapBoundaryCount

`public int` **`getWrapBoundaryCount`**`()`

Returns the number of `wrapBoundary` before wrapping.

Returns:
The count. Default is 1.If < 1 then 1 will be set silently.

---

setWrapBoundaryCount

`public void` **`setWrapBoundaryCount`**`(int count)`

Sets the number of `wrapBoundary` before wrapping.

Parameters:
`count` - The count. Default is 1.If < 1 then 1 will be set silently.

---

getPrimaryDimensionCellTypeCount

`public int` **`getPrimaryDimensionCellTypeCount`**`()`

Returns how may cells in the primary dimension. Unly used if `wrapBoundary` is set.

Returns:
How may cells in the primary dimension. Unly used if `wrapBoundary` is set.

---

setPrimaryDimensionCellTypeCount

`public void` **`setPrimaryDimensionCellTypeCount`**`(int count)`

Sets how many `primDimCellSpanType` a cell spanns in the primary dimension. E.g. `30` (if `primDimCellSpanType` is minutes).

Parameters:
`count` - how many `primDimCellSpanType` a cell spanns in the primary dimension.

---

getPrimaryDimensionCellType

`public int` **`getPrimaryDimensionCellType`**`()`

Returns the type of `primDimCellSpanCount`. E.g `RANGE_TYPE_MINUTE`

Returns:
The type of `primDimCellSpanCount`. E.g `RANGE_TYPE_MINUTE`

---

setPrimaryDimensionCellType

`public void` **`setPrimaryDimensionCellType`**`(int cellType)`

Sets the type of `primDimCellSpanCount`. E.g `RANGE_TYPE_MINUTE`

Parameters:
`cellType` - The type of `primDimCellSpanCount`. E.g `RANGE_TYPE_MINUTE`

---

getPrimaryDimension

`public int` **`getPrimaryDimension`**`()`

(continued from last page)

What dimension time progresses if there was no wrapping. `SwingConstants.HORIZONTAL` or `SwingConstants.VERTICAL`.

Returns:
What dimension time progresses if there was no wrapping.

---

setPrimaryDimension

`public void setPrimaryDimension(int dim)`

Sets the dimension time progresses if there was no wrapping. `SwingConstants.HORIZONTAL` or `SwingConstants.VERTICAL`.

Parameters:
`dim` - What dimension time progresses if there was no wrapping.

---

getActivityLayouts

`public ActivityLayout[] getActivityLayouts()`

Returns the currently installed `com.miginfocom.calendar.layout.ActivityLayouts`.

Returns:
The currently installed `com.miginfocom.calendar.layout.ActivityLayouts`.

See Also:
`addActivityLayout(com.miginfocom.calendar.layout.ActivityLayout)`

---

setActivityLayouts

`public void setActivityLayouts(ActivityLayout[] layouts)`

Sets the `com.miginfocom.calendar.layout.ActivityLayouts`.

Parameters:
`layouts` - The new `com.miginfocom.calendar.layout.ActivityLayouts`.

See Also:
`addActivityLayout(com.miginfocom.calendar.layout.ActivityLayout)`

---

getSnapToMillis

`public long getSnapToMillis()`

Return what even millisecond to snap to when resizing and moving activities.

Returns:
The millisecond to snap to in milliseconds or 0 if snap is disabled.

---

setSnapToMillis

`public void setSnapToMillis(long millis)`

Sets what even millisecond to snap to when resizing and moving activities.

Parameters:
`millis` - The millisecond to snap to in milliseconds. 0 disables snap.

getCategoryHeader

`public CategoryHeaderBean getCategoryHeader()`

Returns the one and only category header bean. It will always be above or to the left of the date area and always on the left side if the primary dimension is horizontal and above if primary dimension is vertical.

Returns:
The one and only category header bean or `null` if none is set.

setCategoryHeader

`public void setCategoryHeader(CategoryHeaderBean header)`

Sets or disables the category header.

Parameters:
`header` - the new header or `null` if it should not be used (i.e. removed).

See Also:
`getCategoryHeader()`

getNorthDateHeader

`public DateHeaderBean getNorthDateHeader()`

Returns the date header that is above the date area.

Returns:
The date header that is above the date area or `null` if none.

setNorthDateHeader

`public void setNorthDateHeader(DateHeaderBean header)`

Sets the date header that will be above the date area.

Parameters:
`header` - The date header that will be above the date area or `null` if none.

getWestDateHeader

`public DateHeaderBean getWestDateHeader()`

Returns the date header that is to the left the date area.

Returns:
The date header that is to the left the date area or `null` if none.

setWestDateHeader

`public void setWestDateHeader(DateHeaderBean header)`

Sets the date header that will be to the left the date area.

Parameters:
`header` - The date header that will be to the left the date area or `null` if none.

getSouthDateHeader

`public DateHeaderBean getSouthDateHeader()`

Returns the date header that is below the date area.

Returns:
The date header that is below the date area or `null` if none.

setSouthDateHeader

`public void setSouthDateHeader(DateHeaderBean header)`

Sets the date header that will be below the date area.

Parameters:
`header` - The date header that will be below the date area or `null` if none.

getEastDateHeader

`public DateHeaderBean getEastDateHeader()`

Returns the date header that is to the right the date area.

Returns:
The date header that is to the right the date area or `null` if none.

setEastDateHeader

`public void setEastDateHeader(DateHeaderBean header)`

Sets the date header that will be to the right the date area.

Parameters:
`header` - The date header that will be to the right the date area or `null` if none.

validateHeaders

`public void validateHeaders()`

getLabelFont

`public java.awt.Font getLabelFont()`

Property: The font used to draw the label.

This is a property that affects the visual representation of the label in the cells.

Returns:
The font used to draw the label. Might be `null`.

See Also:
`setLabelFont(Font)`

(continued from last page)

setLabelFont

```
public void setLabelFont(java.awt.Font font)
```

Property: The font used to draw the label.

This is a property that affects the visual representation of the label in the cells.

Parameters:
`font` - The font used to draw the label. Might be `null`.

See Also:
`getLabelFont()`

---

getLabelNowFont

```
public java.awt.Font getLabelNowFont()
```

Property: The font used to draw the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells.

Returns:
The font used to draw the label. Might be `null`.

See Also:
`setLabelNowFont(Font)`

---

setLabelNowFont

```
public void setLabelNowFont(java.awt.Font font)
```

Property: The font used to draw the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells.

Parameters:
`font` - The font used to draw the label. Might be `null`.

See Also:
`getLabelNowFont()`

---

getLabelAntiAlias

```
public int getLabelAntiAlias()
```

Property: The anti aliasing hint used when drawing the label. For instance `AA_HINT_ON`.

Returns:
The current hint. May be `null`.

See Also:
`confAntiAliasingHint`

---

setLabelAntiAlias

```
public void setLabelAntiAlias(int hint)
```

Property: The anti aliasing hint used when drawing the label. For instance `AA_HINT_ON`.

(continued from last page)

Parameters:
`hint` - The new hint.

See Also:
`confAntiAliasingHint`

---

getLabelBackground

`public java.awt.Paint getLabelBackground()`

Property: The `java.awt.Paint` used to draw the background of the label. It will draw the whole bounds that is specified with `setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)`.

This is a property that affects the visual representation of the label in the cells.

Returns:
The paint used to draw the background of the label. If `null` no background will be painted.

See Also:
`setLabelBackground(Paint)`

---

setLabelBackground

`public void setLabelBackground(java.awt.Paint paint)`

Property: The `java.awt.Paint` used to draw the background of the label. It will draw the whole bounds that is specified with `setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)`.

This is a property that affects the visual representation of the label in the cells.

Parameters:
`paint` - The paint. May be `null`.

See Also:
`getLabelBackground()`

---

getLabelNowBackground

`public java.awt.Paint getLabelNowBackground()`

Property: The `java.awt.Paint` used to draw the background of the label for the cell that spans the current time. It will draw the whole bounds that is specified with `setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)`.

This is a property that affects the visual representation of the label in the cells.

Returns:
The paint used to draw the background of the label. If `null` no background will be painted.

See Also:
`setLabelBackground(Paint)`

---

setLabelNowBackground

`public void setLabelNowBackground(java.awt.Paint paint)`

Property: The `java.awt.Paint` used to draw the background of the label for the cell that spans the current time. It will draw the whole bounds that is specified with `setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)`.

This is a property that affects the visual representation of the label in the cells.

(continued from last page)

Parameters:
`paint` - The paint. May be `null`.

See Also:
`getLabelBackground()`

---

getLabelBorder

`public javax.swing.border.Border getLabelBorder()`

Property: The border that is painted around the label.

This is a property that affects the visual representation of the label in the cells.

Returns:
The border that is painted around the label. `null` means that the label will not be drawn at all, but the background might

See Also:
`setLabelBorder(Border)`

---

setLabelBorder

`public void setLabelBorder(javax.swing.border.Border b)`

Property: The border that is painted around the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:
`b` - The border that is painted around the label. May be `null`.

See Also:
`getLabelBorder()`

---

getLabelNowBorder

`public javax.swing.border.Border getLabelNowBorder()`

Property: The border that is painted around the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells.

Returns:
The border that is painted around the label. `null` means that the label will not be drawn at all, but the background might still be for instance.

See Also:
`setLabelBorder(Border)`

---

setLabelNowBorder

`public void setLabelNowBorder(javax.swing.border.Border b)`

Property: The border that is painted around the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

(continued from last page)

Parameters:

b - The border that is painted around the label. May be `null`.

See Also:

`getLabelBorder()`

---

getLabelForeground

```
public java.awt.Color getLabelForeground()
```

Property: The paint that is used to draw the label.

This is a property that affects the visual representation of the label in the cells.

Returns:

The `java.awt.Paint` used to draw the label. `null` means that the label will not be drawn at all, but the background might still be for instance.

See Also:

`setLabelForeground(Color)`

---

setLabelForeground

```
public void setLabelForeground(java.awt.Color c)
```

Property: The paint that is used to draw the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:

c - The color that is used to draw the label. May be `null`.

See Also:

`getLabelForeground()`

---

getLabelNowForeground

```
public java.awt.Color getLabelNowForeground()
```

Property: The paint that is used to draw the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells.

Returns:

The `java.awt.Paint` used to draw the label. `null` means that the label will not be drawn at all, but the background might still be for instance.

See Also:

`setLabelForeground(Color)`

---

setLabelNowForeground

```
public void setLabelNowForeground(java.awt.Color c)
```

Property: The paint that is used to draw the label for the cell that spans the current time.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:

(continued from last page)

c - The color that is used to draw the label. May be null.

See Also:
getLabelForeground()

---

getLabelPlaceRect

public PlaceRect **getLabelPlaceRect**()

Property: Denotes the bounds that the label should get. The PlaceRect will be given the bounds of the cell and return/convert that to the bounds that the label should have.

This is a property that affects the visual representation of the label in the cells. Setting the value to null will normally make this label not appear at all.

Returns:
The specifivation

See Also:
setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)

---

setLabelPlaceRect

public void **setLabelPlaceRect**(PlaceRect placeRect)

Property: Denotes the bounds that the label should get. The PlaceRect object will be given the bounds of the cell and return/convert that to the bounds that the label should have.

This is a property that affects the visual representation of the label in the cells. Setting the value to null will normally make this label not appear at all.

Parameters:
placeRect - The new bounds spec. May be null.

See Also:
getLabelPlaceRect()

---

getLabelAlignX

public AtRefRangeNumber **getLabelAlignX**()

Property: The horizontal alignment the label should have within the bounds denoted by the PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect)). It can be a fixed number of pixels from either end or relative (say 50% in, which means centered). There are a number of subclasses to this interface such as com.miginfocom.util.gfx.geometry.numbers.AtStart, com.miginfocom.util.gfx.geometry.numbers.AtEnd and com.miginfocom.util.gfx.geometry.numbers.AtFraction.

E.g. new AtFraction(0.5f) or new AtStart(10f).

This is a property that affects the visual representation of the label in the cells.

Returns:
The current alignment. Not null.

See Also:
setLabelAlignX(com.miginfocom.util.gfx.geometry.numbers.AtRefRangeNumber)

---

setLabelAlignX

public void **setLabelAlignX**(AtRefRangeNumber x)

(continued from last page)

Property: The horizontal alignment the label should have within the bounds denoted by the `PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect))`. It can be a fixed number of pixels from either end or relative (say 50% in, which means centered). There are a number of subclasses to this interface such as `com.miginfocom.util.gfx.geometry.numbers.AtStart`, `com.miginfocom.util.gfx.geometry.numbers.AtEnd` and `com.miginfocom.util.gfx.geometry.numbers.AtFraction`.

E.g. `new AtFraction(0.5f)` or `new AtStart(10f)`.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:

`x` - The new horizontal alignment. Not `null`.

See Also:

`getLabelAlignX()`

---

getLabelAlignY

```
public AtRefRangeNumber getLabelAlignY()
```

Property: The vertical alignment the label should have within the bounds denoted by the `PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect))`. It can be a fixed number of pixels from either end or relative (say 50% in, which means centered). There are a number of subclasses to this interface such as `com.miginfocom.util.gfx.geometry.numbers.AtStart`, `com.miginfocom.util.gfx.geometry.numbers.AtEnd` and `com.miginfocom.util.gfx.geometry.numbers.AtFraction`.

E.g. `new AtFraction(0.5f)` or `new AtStart(10f)`.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Returns:

The current alignment. Not `null`.

See Also:

`setLabelAlignY(com.miginfocom.util.gfx.geometry.numbers.AtRefRangeNumber)`

---

setLabelAlignY

```
public void setLabelAlignY(AtRefRangeNumber y)
```

Property: The vertical alignment the label should have within the bounds denoted by the `PlaceRact (setLabelPlaceRect(com.miginfocom.util.gfx.geometry.PlaceRect))`. It can be a fixed number of pixels from either end or relative (say 50% in, which means centered). There are a number of subclasses to this interface such as `com.miginfocom.util.gfx.geometry.numbers.AtStart`, `com.miginfocom.util.gfx.geometry.numbers.AtEnd` and `com.miginfocom.util.gfx.geometry.numbers.AtFraction`.

E.g. `new AtFraction(0.5f)` or `new AtStart(10f)`.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:

`y` - The new vertical alignment. Not `null`.

See Also:

`getLabelAlignY()`

(continued from last page)

getLabelDateFormat

`public String getLabelDateFormat()`

Property: The date format that specifies the text that will be the label. The start date/time that the cell spans will be formatted with this date format and the result will be the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Returns:
The current date format. May be `null`.

See Also:
`setLabelDateFormat(String)`

---

setLabelDateFormat

`public void setLabelDateFormat(String format)`

Property: The date format that specifies the text that will be the label. The start date/time that the cell spans will be formatted with this date format and the result will be the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:
`format` - The new date format. May be `null` which will make the label not show.

See Also:
`getLabelDateFormat()`

---

getLabelNowDateFormat

`public String getLabelNowDateFormat()`

Property: The date format that specifies the text that will be the label for the cell that spans the current time. The start date/time that the cell spans will be formatted with this date format and the result will be the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Returns:
The current date format. May be `null`.

See Also:
`setLabelNowDateFormat(String)`

---

setLabelNowDateFormat

`public void setLabelNowDateFormat(String format)`

Property: The date format that specifies the text that will be the label for the cell that spans the current time. The start date/time that the cell spans will be formatted with this date format and the result will be the label.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:
`format` - The new date format. May be `null` which will make the label not show.

(continued from last page)

See Also:
`getLabelNowDateFormat()`

---

getLabelNowRangeType

`public Integer getLabelNowRangeType()`

Property: The range type to round the "now" date to. Can for instance be set to `RANGE_TYPE_MONTH` to make all the cells in a month be cosidered "now" and get the "now" font/background/foreground.

Returns:
The current range type or `null` if not set. E.g. `DateRangeI.RANGE_TYPE_MONTH`.

---

setLabelNowRangeType

`public void setLabelNowRangeType(Integer rangeType)`

Property: The range type to round the "now" date to. Can for instance be set to `RANGE_TYPE_MONTH` to make all the cells in a month be cosidered "now" and get the "now" font/background/foreground.

Parameters:
`rangeType` - E.g. `DateRangeI.RANGE_TYPE_MONTH`.

---

getLabelFirstDateFormat

`public String getLabelFirstDateFormat()`

Property: Date format for the "first" of something, e.g. first day of the month.

Returns:
Date format for the "first" of something, e.g. first day of the month.

Since:
6.0

---

setLabelFirstDateFormat

`public void setLabelFirstDateFormat(String format)`

Property: Date format for the "first" of something, e.g. first day of the month.

Parameters:
`format` - Date format for the "first" of something, e.g. first day of the month.

Since:
6.0

---

getLabelFirstInField

`public int getLabelFirstInField()`

Property: If `getFirstDateFormat()` is != this is the `java.util.Calendar` field that is used to know what's "first".

Returns:
The range type. E.g. `Calendar.DAY_OF_MONTH`.

Since:
6.0

setLabelFirstInField

`public void setLabelFirstInField(int calField)`

Property: If `getFirstDateFormat()` is != this is the `java.util.Calendar` field that is used to know what's "first".

Parameters:
`calField` - The range type. E.g. `Calendar.DAY_OF_MONTH`.

Since:
6.0

getLabelCellModulo

`public int getLabelCellModulo()`

Property: This property make is possible to show every second or third (for instance) cell label instead if the default every one. A value of 2 will show every second label etc.

This is a property that affects the visual representation of the label in the cells.

Returns:
The current modulo.

See Also:
`setLabelCellModulo(int)`

setLabelCellModulo

`public void setLabelCellModulo(int i)`

Property: This property make is possible to show every second or third (for instance) cell label instead if the default every one. A value of 2 will show every second label etc.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:
`i` - The new modulo. 1 is default and wil paint every one.

See Also:
`getLabelCellModulo()`

getLabelMinimumCellSize

`public java.awt.Dimension getLabelMinimumCellSize()`

Property: If the cell is smaller than this size, in any dimension, the label will not be drawn. This can be used to hide the label if the cells get to be too small to show the label in an acceptable way.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Returns:
The current minimum cell size. May be `null`.

See Also:
`setLabelMinimumCellSize(Dimension)`

setLabelMinimumCellSize

`public void setLabelMinimumCellSize(java.awt.Dimension size)`

Property: If the cell is smaller than this size, in any dimension, the label will not be drawn. This can be used to hide the label if the cells get to be too small to show the label in an acceptable way.

This is a property that affects the visual representation of the label in the cells. Setting the value to `null` will normally make this label not appear at all.

Parameters:
`size` - The new minumum size. May be `null` in which case the label is always shown.

See Also:
`getLabelMinimumCellSize()`

getNoExpandedFolderGridLine

`public boolean getNoExpandedFolderGridLine()`

Property: If the grid lines around a folded (collapsed) folder sub row should be merged to one. This is to avoid the double grid lines that would be visible around rows with size 0 otherwise. It is purely a visual setting and won't affect any data or functionality.

Returns:
The current state.

setNoExpandedFolderGridLine

`public void setNoExpandedFolderGridLine(boolean b)`

Property: If the grid lines around a folded (collapsed) folder sub row should be merged to one. This is to avoid the double grid lines that would be visible around rows with size 0 otherwise. It is purely a visual setting and won't affect any data or functionality.

Parameters:
`b` - The new state. Default if `false`.

getHorizontalGridLinePaintEven

`public java.awt.Paint getHorizontalGridLinePaintEven()`

Property: The paint used to draw the even (0, 2, 4...) horizontal grid lines.

Returns:
The current paint. May be `null`.

setHorizontalGridLinePaintEven

`public void setHorizontalGridLinePaintEven(java.awt.Paint paint)`

Property: The paint used to draw the even (0, 2, 4...) horizontal grid lines.

Parameters:
`paint` - The new paint. May be `null`.

(continued from last page)

getHorizontalGridLinePaintOdd

`public java.awt.Paint getHorizontalGridLinePaintOdd()`

Property: The paint used to draw the odd (1, 3, 5...) horizontal grid lines.

Returns:
The current paint. May be `null`.

---

setHorizontalGridLinePaintOdd

`public void setHorizontalGridLinePaintOdd(java.awt.Paint paint)`

Property: The paint used to draw the odd (1, 3, 5...) horizontal grid lines.

Parameters:
`paint` - The new paint. May be `null`.

---

getVerticalGridLinePaintEven

`public java.awt.Paint getVerticalGridLinePaintEven()`

Property: The paint used to draw the even (0, 2, 4...) vertical grid lines.

Returns:
The current paint. May be `null`.

---

setVerticalGridLinePaintEven

`public void setVerticalGridLinePaintEven(java.awt.Paint paint)`

Property: The paint used to draw the even (0, 2, 4...) vertical grid lines.

Parameters:
`paint` - The new paint. May be `null`.

---

getVerticalGridLinePaintOdd

`public java.awt.Paint getVerticalGridLinePaintOdd()`

Property: The paint used to draw the odd (1, 3, 5...) vertical grid lines.

Returns:
The current paint. May be `null`.

---

setVerticalGridLinePaintOdd

`public void setVerticalGridLinePaintOdd(java.awt.Paint paint)`

Property: The paint used to draw the odd (1, 3, 5...) vertical grid lines.

Parameters:
`paint` - The new paint. May be `null`.

---

getSubRowGridLinePaint

`public java.awt.Paint getSubRowGridLinePaint()`

(continued from last page)

Property: The paint used to draw sub row divider grid lines. This will only be in the secondary dimension since sub rows can only exist in the secondary dimension.

Returns:

The current paint. May be `null`.

---

setSubRowGridLinePaint

```
public void setSubRowGridLinePaint(java.awt.Paint paint)
```

Property: The paint used to draw sub row divider grid lines. This will only be in the secondary dimension since sub rows can only exist in the secondary dimension.

Parameters:

`paint` - The new paint. may be `null`.

---

getHorizontalGridLineExceptions

```
public GridLineException[] getHorizontalGridLineExceptions()
```

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Returns:

The exceptions or zero length if no exteptions to the grid lines exist (default). Never `null`.

---

setHorizontalGridLineExceptions

```
public void setHorizontalGridLineExceptions(GridLineException[] exceptions)
```

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Parameters:

`exceptions` - The exceptions or zero length if no exteptions to the grid lines exist (default). Never `null`.

---

getVerticalGridLineExceptions

```
public GridLineException[] getVerticalGridLineExceptions()
```

(continued from last page)

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Returns:

The exceptions or zero length if no exteptions to the grid lines exist (default). Never `null`.

---

setVerticalGridLineExceptions

```
public void setVerticalGridLineExceptions(GridLineException[] exceptions)
```

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Parameters:

`exceptions` - The exceptions or zero length if no exteptions to the grid lines exist (default). Never `null`.

---

getSelectionMouseOverPaint

```
public java.awt.Paint getSelectionMouseOverPaint()
```

Property: The paint used to fill the cells that the mouse is over. `null` indicated no mouse over painting.

Returns:

The current paint. May be `null`.

See Also:

`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

---

setSelectionMouseOverPaint

```
public void setSelectionMouseOverPaint(java.awt.Paint paint)
```

Property: The paint used to fill the cells that the mouse is over. `null` indicated no mouse over painting.

Parameters:

`paint` - The new paint. May be `null`.

See Also:

`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

getSelectionMousePressedPaint

`public java.awt.Paint getSelectionMousePressedPaint()`

Property: The paint used to fill the cells that the is pressed over (e.g. temporary selection). `null` indicated no mouse press painting.

Returns:
The current paint. May be `null`.

See Also:
`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

setSelectionMousePressedPaint

`public void setSelectionMousePressedPaint(java.awt.Paint paint)`

Property: The paint used to fill the cells that the is pressed over (e.g. temporary selection). `null` indicated no mouse press painting.

Parameters:
`paint` - The new paint. May be `null`.

See Also:
`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

getSelectionPaint

`public java.awt.Paint getSelectionPaint()`

Property: The paint used to fill the cells that are selected. `null` indicated no selection painting.

Returns:
The current paint. May be `null`.

See Also:
`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

setSelectionPaint

`public void setSelectionPaint(java.awt.Paint paint)`

Property: The paint used to fill the cells that are selected. `null` indicated no selection painting.

Parameters:
`paint` - The new paint. May be `null`.

See Also:
`setDateRangeSelection(com.miginfocom.util.dates.DateRangeI, int, boolean)`

getSelectablePaint

`public java.awt.Paint getSelectablePaint()`

Property: The paint used to fill the cells that are marked as selectable. `null` indicated no selectable painting.

Returns:
The current paint. May be `null`.

(continued from last page)

See Also:
`setSelectableRange(com.miginfocom.util.dates.DateRangeI)`

---

setSelectablePaint

`public void setSelectablePaint(java.awt.Paint paint)`

Property: The paint used to fill the cells that are marked as selectable. `null` indicated no selectable painting.

Parameters:
`paint` - The new paint. May be `null`.

See Also:
`setSelectableRange(com.miginfocom.util.dates.DateRangeI)`

---

getSelectionBoundaryType

`public int getSelectionBoundaryType()`

Property: The type of date range (e.g. day, hour, week) that selectionw will be rounded to. For instance if set to `RANGE_TYPE_DAY` only full days can be selected, one can not select just a couple of hours.

Returns:
The current boundary type. E.g. `DateRangeI.RANGE_TYPE_WEEK`.

---

setSelectionBoundaryType

`public void setSelectionBoundaryType(int rangeType)`

Property: The type of date range (e.g. day, hour, week) that selectionw will be rounded to. For instance if set to `RANGE_TYPE_DAY` only full days can be selected, one can not select just a couple of hours.

Parameters:
`rangeType` - The new boundary type. E.g. `DateRangeI.RANGE_TYPE_HOUR`.

---

getOddRowPaint

`public java.awt.Paint getOddRowPaint()`

Property: The paint to paint for all odd rows. May be `null`.

Returns:
The current paint. May be `null`.

---

setOddRowPaint

`public void setOddRowPaint(java.awt.Paint paint)`

Property: The paint to paint for all odd rows. May be `null`.

Parameters:
`paint` - The new paint. May be `null`.

---

getOddColumnPaint

`public java.awt.Paint getOddColumnPaint()`

Property: The paint to paint for all odd columns. May be `null`.

(continued from last page)

Returns:
The current paint. May be `null`.

setOddColumnPaint

`public void setOddColumnPaint(java.awt.Paint paint)`

Property: The paint to paint for all odd columns. May be `null`.

Parameters:
`paint` - The new paint. May be `null`.

getEvenBoundaryPaint

`public java.awt.Paint getEvenBoundaryPaint()`

Property: The paint used to fill all even (0, 2, 4...) date ranges of the date area. For instance every even month. The odd date ranges will get the background color of the date area since they are not painted at all. Use `getEvenRangeType()` and `getEvenRangeTypeCount()` to set the range type/count.

Returns:
The current paint. May be `null`.

See Also:
`getEvenRangeType()`
`getEvenRangeTypeCount()`

setEvenBoundaryPaint

`public void setEvenBoundaryPaint(java.awt.Paint paint)`

Property: The paint used to fill all even (0, 2, 4...) date ranges of the date area. For instance every even month. The odd date ranges will get the background color of the date area since they are not painted at all. Use `getEvenRangeType()` and `getEvenRangeTypeCount()` to set the range type/count.

Parameters:
`paint` - The new paint. May be `null`.

See Also:
`setEvenRangeType(int)`
`setEvenRangeTypeCount(int)`

getEvenRangeType

`public int getEvenRangeType()`

Property: The date range type (i.e. Unit. E.g. day, week, month) that the paint boundary have. If for instance every second month should be painted in another color this property should be set to `DateRangeI.RANGE_TYPE_MONTH`.

Returns:
The current date range type. E.g. `DateRangeI.RANGE_TYPE_MONTH`.

setEvenRangeType

`public void setEvenRangeType(int rangeType)`

Property: The date range type (i.e. Unit. E.g. day, week, month) that the paint boundary have. If for instance every second month should be painted in another color this property should be set to `DateRangeI.RANGE_TYPE_MONTH`.

(continued from last page)

Parameters:
`rangeType` - The new date range type. E.g. `DateRangeI.RANGE_TYPE_MONTH`.

---

getEvenRangeTypeCount

`public int getEvenRangeTypeCount()`

Property: How many of `getEvenRangeType()` that should be bunched up and be called as one. If for instance this value is 3 and the type is `DateRangetI.RANGE_TYPE_DAY` then day 0, 1, 2 will be painted with the even paint. 3, 4, 5 will not be painted. 6, 7, 8 will be painted with the even painte again and so on.

Returns:
The current date range type count.

---

setEvenRangeTypeCount

`public void setEvenRangeTypeCount(int count)`

Property: How many of `getEvenRangeType()` that should be bunched up and be called as one. If for instance this value is 3 and the type is `DateRangetI.RANGE_TYPE_DAY` then day 0, 1, 2 will be painted with the even paint. 3, 4, 5 will not be painted. 6, 7, 8 will be painted with the even painte again and so on.

Parameters:
`count` - The new date range type count.

---

getDividerPaint

`public java.awt.Paint getDividerPaint()`

Property: The paint used to draw separator lines between some date ranges, for instance months.

Returns:
The current paint. May be `null`.

See Also:
`getDividerRangeType()`

---

setDividerPaint

`public void setDividerPaint(java.awt.Paint paint)`

Property: The paint used to draw separator lines between some date ranges, for instance months.

Parameters:
`paint` - The new paint. May be `null`.

See Also:
`setDividerRangeType(int)`

---

getDividerRangeType

`public int getDividerRangeType()`

Property: The type of the divider range (e.g. day, week, month). If the divider should be drawn between months for instance this value should be `RANGE_TYPE_MONTH`.

Returns:
The current divider date range type. E.g. `DateRangeI.RANGE_TYPE_MONTH`.

setDividerRangeType

`public void setDividerRangeType(int rangeType)`

Property: The type of the divider range (e.g. day, week, month). If the divider should be drawn between months for instance this value should be `RANGE_TYPE_MONTH`.

Parameters:
`rangeType` - The new divider date range type. E.g. `DateRangeI.RANGE_TYPE_MONTH`.

getMouseOverActivitiesOntop

`public boolean getMouseOverActivitiesOntop()`

Property: If Activites that have the mouse hovering over them should be painted on top of other activities.

Returns:
The current value. Default is false.

setMouseOverActivitiesOntop

`public void setMouseOverActivitiesOntop(boolean b)`

Property: If Activites that have the mouse hovering over them should be painted on top of other activities.

Parameters:
`b` - The new value. Default is false.

getSelectedActivitiesOntop

`public boolean getSelectedActivitiesOntop()`

Property: If Activites that are selected should be painted on top of other activities.

Returns:
The current value. Default is false.

setSelectedActivitiesOntop

`public void setSelectedActivitiesOntop(boolean b)`

Property: If Activites that are selected should be painted on top of other activities.

Parameters:
`b` - The new value. Default is false.

getActivityDepositoryContext

`public String getActivityDepositoryContext()`

Property: The context used as a key in the `com.miginfocom.calendar.activity.ActivityDepository` to get the activities that this date area should show. In the depository activities can optionally be put in different "containers" which are adressed by this context. The default context is `null`.

Returns:
The current context. Default is `null`.

(continued from last page)

setActivityDepositoryContext

```
public void setActivityDepositoryContext(String context)
```

Property: The context used as a key in the `com.miginfocom.calendar.activity.ActivityDepository` to get the activities that this date area should show. In the depository activities can optionally be put in different "containers" which are adressed by this context. The default context is `null`.

Parameters:

`context` - The new context. Default is `null`.

---

isShowNoFitIcon

```
public boolean isShowNoFitIcon()
```

Property: If the little icon that shows that not all activities have fitted within a date range (normally a day). Mouse presses on this icon, as well as how the icon/AShape should look like is handled by the `DefaultDateArea` that this JavBean aggregates. It is gettable by `getDefaultDateArea()`.

To register a listener for mouse events register a `com.miginfocom.ashape.interaction.InteractionListener` on the date area.

To change the AShape used for the no fit shape call `setDefaultNoFitShape(com.miginfocom.ashape.shapes.RootAShape)` after calling this method (this method sets it to default).

This method basically set the shape to `null` or `AShapeUtil.createDefaultNoFitShape(DefaultDateArea.NO_FIT_EVENT_PROPERTY)` depending on `b`.

Returns:

If the not fit icon will be shown. Basically checks if the shape is set to `null`.

---

setShowNoFitIcon

```
public void setShowNoFitIcon(boolean b)
```

Property: If the little icon that shows that not all activities have fitted within a date range (normally a day). Mouse presses on this icon, as well as how the icon/AShape should look like is handled by the `DefaultDateArea` that this JavBean aggregates. It is gettable by `getDefaultDateArea()`.

To register a listener for mouse events register a `com.miginfocom.ashape.interaction.InteractionListener` on the date area.

To change the AShape used for the no fit shape call `setDefaultNoFitShape(com.miginfocom.ashape.shapes.RootAShape)` after calling this method (this method sets it to default).

Parameters:

`b` - If the not fit icon will be shown. Basically checks if the shape is set to `null`.

---

getSelectionType

```
public int getSelectionType()
```

Property: The type of selection that can be done on the date area background. This has nothing to do with activity selection.

Returns:

The old selection type. `DateArea.SELECTION_TYPE_NORMAL` or `DateArea.SELECTION_TYPE_NONE`.

(continued from last page)

setSelectionType

`public void setSelectionType(int type)`

Property: The type of selection that can be done on the date area background. This has nothing to do with activity selection.

Parameters:

`type` - If the date area cells can be selected or not. `DateArea.SELECTION_TYPE_NORMAL` or `DateArea.SELECTION_TYPE_NONE`.

---

getBackgroundPaint

`public java.awt.Paint getBackgroundPaint()`

Property: The background paint in the date area. Will override the normal background `Color` so that `Paint` objects can be used instead.

Returns:

The current background paint. May be `null`.

---

setBackgroundPaint

`public void setBackgroundPaint(java.awt.Paint p)`

Property: The background paint in the date area. Will override the normal background `Color` so that `Paint` objects can be used instead.

Parameters:

`p` - The new background paint. May be `null`.

---

getDateAreaInnerBorder

`public javax.swing.border.Border getDateAreaInnerBorder()`

Property: The inner border around the date area. The border will be set on the view so that parts of it will be covered if there is scrolling.

Returns:

The current border. May be `null`.

See Also:

`getDateAreaOuterBorder()`

---

setDateAreaInnerBorder

`public void setDateAreaInnerBorder(javax.swing.border.Border b)`

Property: The inner border around the date area. The border will be set on the view so that parts of it will be covered if there is scrolling.

Parameters:

`b` - The new border. May be `null`.

See Also:

`setDateAreaOuterBorder(Border)`

---

getDateAreaOuterBorder

`public javax.swing.border.Border getDateAreaOuterBorder()`

(continued from last page)

Property: The outer border around the date area. The border will be installed on the scroll pane so that it will normally always be shown fully even if the view is partly scrolled away (invisible).

Returns:

The current border. May be `null`.

See Also:

`getDateAreaInnerBorder()`

---

setDateAreaOuterBorder

```
public void setDateAreaOuterBorder(javax.swing.border.Border b)
```

Property: The outer border around the date area. The border will be installed on the scroll pane so that it will normally always be shown fully even if the view is partly scrolled away (invisible).

Parameters:

`b` - The new border. May be `null`.

See Also:

`setDateAreaInnerBorder(Border)`

---

addActivityDragResizeListener

```
public void addActivityDragResizeListener(ActivityDragResizeListener l)
```

This call is transmitted to the `addActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener)`

---

addActivityDragResizeListener

```
public void addActivityDragResizeListener(ActivityDragResizeListener l,
          boolean asWeakRef)
```

This call is transmitted to the `addActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener, boolean)`

---

removeActivityDragResizeListener

```
public void removeActivityDragResizeListener(ActivityDragResizeListener l)
```

This call is transmitted to the `removeActivityDragResizeListener(com.miginfocom.calendar.datearea.ActivityDragResizeListener)`

---

addDateChangeListener

```
public void addDateChangeListener(DateChangeListener l)
```

This call is transmitted to the `addDateChangeListener(com.miginfocom.util.dates.DateChangeListener)`

---

addDateChangeListener

```
public void addDateChangeListener(DateChangeListener l,
          boolean asWeakRef)
```

This call is transmitted to the `addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)`

removeDateChangeListener

```
public void removeDateChangeListener(DateChangeListener l)
```

This call is transmitted to the `removeDateChangeListener(com.miginfocom.util.dates.DateChangeListener)`

addInteractionListener

```
public void addInteractionListener(InteractionListener l)
```

This call is transmitted to the `addInteractionListener(com.miginfocom.ashape.interaction.InteractionListener)`

addInteractionListener

```
public void addInteractionListener(InteractionListener l,
         boolean asWeakRef)
```

This call is transmitted to the `addInteractionListener(com.miginfocom.ashape.interaction.InteractionListener, boolean)`

removeInteractionListener

```
public void removeInteractionListener(InteractionListener l)
```

This call is transmitted to the `removeInteractionListener(com.miginfocom.ashape.interaction.InteractionListener)`.

addActivityMoveListener

```
public void addActivityMoveListener(ActivityMoveListener l)
```

This call is transmitted to the `addActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener)`

addActivityMoveListener

```
public void addActivityMoveListener(ActivityMoveListener l,
         boolean asWeakRef)
```

This call is transmitted to the `addActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener, boolean)`

removeActivityMoveListener

```
public void removeActivityMoveListener(ActivityMoveListener l)
```

This call is transmitted to the `removeActivityMoveListener(com.miginfocom.calendar.datearea.ActivityMoveListener)`

com.miginfocom.beans
Class DateComboBean

```
java.lang.Object
   |-ComboDateSpinner
      |-com.miginfocom.beans.DateComboBean
```

public class **DateComboBean**
extends ComboDateSpinner

Constructor Summary

public	DateComboBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

DateComboBean

`public` **`DateComboBean`**`()`

com.miginfocom.beans
Class DateGroupConnectorBean

```
java.lang.Object
   ┗-DateSpinnerGroup
      ┗-com.miginfocom.beans.DateGroupConnectorBean
```

public class **DateGroupConnectorBean**
extends DateSpinnerGroup

A `com.miginfocom.calendar.spinner.DateSpinnerGroup` that has this sub class only to be a JAvaBean

Constructor Summary

public	DateGroupConnectorBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

DateGroupConnectorBean

`public` **`DateGroupConnectorBean`**`()`

com.miginfocom.beans
Class DateHeaderBean

```
java.lang.Object
   |-com.miginfocom.beans.AbstractBean
      |-com.miginfocom.beans.AbstractHeaderBean
         |-com.miginfocom.beans.DateHeaderBean
```

All Implemented Interfaces:
Serializable

---

public class **DateHeaderBean**
extends AbstractHeaderBean

A header object that wraps (aggregates) a real `com.miginfocom.calendar.header.Header` implementation.

This bean is for simplifying the usage of a header and to make the header more JavaBean-like.

Constructor Summary

public	DateHeaderBean()

Method Summary

void	addDateChangeListener(DateChangeListener l) This call is transmitted to the addDateChangeListener(com.miginfocom.util.dates.DateChangeListener) Note that mouse effects must be enabled.
void	addDateChangeListener(DateChangeListener l, boolean asWeakRef) This call is transmitted to the addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean) Note that mouse effects must be enabled.
java.awt.Color	getGridColor() Property: The color of the grid lines.
GridLineException[]	getGridLineExceptions() Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
Header	getHeader()
CellDecorationRow[]	getHeaderRows() Property: An array of rows that makes out the different rows or colums for the header (rows for north/south headers and colums for east/west headers).
javax.swing.border.Border	getInnerBorder() Property: The inner border around the header.
int	getLabelRotation() Property: If the labels for the header should be rotated and if so how.

String	getLabelRotationRows() Property: A comma separated string with the rows that are to get the property getLabelRotation().
boolean	getMouseEffectsEnabled() Property: If mouse listeners should be installed on the header to make live effects enabled on the cell rows.
javax.swing.border.Border	getOuterBorder() Property: The outer border around the header.
int	getRowGap() The gap in pixels between the header rows.
int	getTextAntiAlias() Property: The text antialias hint that will be set on the header's renderer before any text is drawn.
String	getTextAntiAliasRows() Property: A comma separated string with the rows that are to get the property getTextAntiAlias().
void	removeDateChangeListener(DateChangeListener l) This call is transmitted to the removeDateChangeListener(com.miginfocom.util.dates.DateChangeListener)
void	setGridColor(java.awt.Color c) Property: The color of the grid lines.
void	setGridLineExceptions(GridLineException[] exceptions) Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way.
void	setHeaderRows(CellDecorationRow[] rows) Property: An array of rows that makes out the different rows or colums for the header (rows for north/south headers and colums for east/west headers).
void	setInnerBorder(javax.swing.border.Border border) Property: The inner border around the header.
void	setLabelRotation(int rot) Property: If the labels for the header should be rotated and if so how.
void	setLabelRotationRows(String rows) Property: A comma separated string with the rows that are to get the property getLabelRotation().
void	setMouseEffectsEnabled(boolean b) Property: If mouse listeners should be installed on the header to make live effects enabled on the cell rows.
void	setOuterBorder(javax.swing.border.Border border) Property: The outer border around the header.
void	setRowGap(int rowGap) The gap in pixels between the header rows.
void	setTextAntiAlias(int hint) Property: The text antialias hint that will be set on the header's renderer before any text is drawn.

void	setTextAntiAliasRows(String rows) Property: A comma separated string with the rows that are to get the property getTextAntiAlias().

Methods inherited from class com.miginfocom.beans.AbstractHeaderBean

getBackgroundPaint, getContainer, getEdge, getExpandToCorner, isVisible, revalidateRepaintContainer, setBackgroundPaint, setDateAreaContainer, setEdge, setExpandToCorner, setVisible

Methods inherited from class com.miginfocom.beans.AbstractBean

addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

DateHeaderBean

public **DateHeaderBean**()

Methods

getHeader

public Header **getHeader**()

getGridLineExceptions

public GridLineException[] **getGridLineExceptions**()

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Returns:

The exceptions or zero length if no exteptions to the grid lines exist (default). Never null.

(continued from last page)

setGridLineExceptions

`public void setGridLineExceptions(GridLineException[] exceptions)`

Property: Grid line exceptions makes it very fleible to override the gridline size and color in a structured way. Examples of exceptions is:

- Every fourth grid line.
- Every grid line that is on a day boundary (any date range type).
- Every grid line on an hour boundary but only for hours 8 to 16.

You can have several exceptions (as this property is an array) each capable of the above list and if the first exception is not a "hit" then the next one will be evaluated and so on until the array is finnished and the grid line used will be the default one.

To really understand how flexible and powerful this feature is see the "Repetitions" section in the FAQ document.

Parameters:

`exceptions` - The exceptions or zero length if no exteptions to the grid lines exist (default). Never `null`.

---

getLabelRotation

`public int getLabelRotation()`

Property: If the labels for the header should be rotated and if so how. The rotaton can be either:

1. `TYPE_SINGLE_LINE_ROT_CW` (right 90 degrees)
2. `TYPE_SINGLE_LINE_ROT_CCW` (left 90 degrees)
3. `TYPE_SINGLE_LINE` (normal text)

Returns:

The rotation. Default is `TextAShape.TYPE_SINGLE_LINE` which means no rotation.

---

setLabelRotation

`public void setLabelRotation(int rot)`

Property: If the labels for the header should be rotated and if so how. The rotaton can be either:

1. `TYPE_SINGLE_LINE_ROT_CW` (right 90 degrees)
2. `TYPE_SINGLE_LINE_ROT_CCW` (left 90 degrees)
3. `TYPE_SINGLE_LINE` (normal text)

This method will reinitialize the header if the value is changed but do nothing if it's the same.

Parameters:

`rot` - The rotation. Default is `TextAShape.TYPE_SINGLE_LINE` which means no rotation.

---

getLabelRotationRows

`public String getLabelRotationRows()`

Property: A comma separated string with the rows that are to get the property `getLabelRotation()`. E.g. "1,2,4". A value of `null`, "" and any illegal value means "all rows".

Returns:

(continued from last page)

The current value.

setLabelRotationRows

`public void setLabelRotationRows(String rows)`

Property: A comma separated string with the rows that are to get the property `getLabelRotation()`. E.g. "1,2,4". A value of `null`, "" and any illegal value means "all rows".

This method will reinitialize the header if the value is changed but do nothing if it's the same.

Parameters:

`rows` - The new value.

getTextAntiAlias

`public int getTextAntiAlias()`

Property: The text antialias hint that will be set on the header's renderer before any text is drawn. If the value is `GfxUtil.AA_HINT_INHERIT` the value will not be changed and thus the default value for the graphics environment will be used.

Normal values are (but other values may apply at a later time, for instance to set sub pixel anti aliasing):

1. `AA_HINT_ON`
2. `AA_HINT_OFF`
3. `AA_HINT_LCD_GASP`

Returns:

The current value.

setTextAntiAlias

`public void setTextAntiAlias(int hint)`

Property: The text antialias hint that will be set on the header's renderer before any text is drawn. If the value is `GfxUtil.AA_HINT_INHERIT` the value will not be changed and thus the default value for the graphics environment will be used.

Normal values are (but other values may apply at a later time, for instance to set sub pixel anti aliasing):

1. `AA_HINT_ON`
2. `AA_HINT_OFF`
3. `AA_HINT_LCD_GASP`

This method will reinitialize the header if the value is changed but do nothing if it's the same.

Parameters:

`hint` - The new value.

getTextAntiAliasRows

`public String getTextAntiAliasRows()`

Property: A comma separated string with the rows that are to get the property `getTextAntiAlias()`. E.g. "1,2,4". A value of `null`, "" and any illegal value means "all rows".

Returns:

(continued from last page)

The current value.

setTextAntiAliasRows

`public void setTextAntiAliasRows(String rows)`

Property: A comma separated string with the rows that are to get the property `getTextAntiAlias()`. E.g. "1,2,4". A value of `null`, "" and any illegal value means "all rows".

This method will reinitialize the header if the value is changed but do nothing if it's the same.

Parameters:

`rows` - The new value.

getGridColor

`public java.awt.Color getGridColor()`

Property: The color of the grid lines. If `null` no lines will be drawn.

Note! The grid color if set will show over everything so that no cells will look merged. To show a grid where cell rows can have cells that span more that one row/column use the background paint of the whole header to set the grid color and leave this grid color to `null`. This will make it possible to create headers like in `getHeaderRows()` JavDoc. Also see `AbstractHeaderBean.setBackgroundPaint(Paint)`

Returns:

The current grid line color. May be `null`.

setGridColor

`public void setGridColor(java.awt.Color c)`

Property: The color of the grid lines. If `null` no lines will be drawn.

Note! The grid color if set will show over everything so that no cells will look merged. To show a grid where cell rows can have cells that span more that one row/column use the background paint of the whole header to set the grid color and leave this grid color to `null`. This will make it possible to create headers like in `getHeaderRows()` JavDoc. Also see `AbstractHeaderBean.setBackgroundPaint(Paint)`

Parameters:

`c` - The new grid line color. May be `null`.

getRowGap

`public int getRowGap()`

The gap in pixels between the header rows.

Returns:

The gap in pixels between the header rows.

Since:

6.0

setRowGap

`public void setRowGap(int rowGap)`

The gap in pixels between the header rows.

Parameters:

(continued from last page)

`rowGap` - The gap in pixels between the header rows.

Since:
6.0

---

getInnerBorder

`public javax.swing.border.Border getInnerBorder()`

Property: The inner border around the header. The border vill be set on the view so that parts of it will be covered if there is scrolling.

Returns:
The current border. May be `null`.

See Also:
`getOuterBorder()`

---

setInnerBorder

`public void setInnerBorder(javax.swing.border.Border border)`

Property: The inner border around the header. The border vill be set on the view so that parts of it will be covered if there is scrolling.

Parameters:
`border` - The new border. May be `null`.

See Also:
`setOuterBorder(Border)`

---

getOuterBorder

`public javax.swing.border.Border getOuterBorder()`

Property: The outer border around the header. The border will be installed on the scroll pane so that it will normally always be shown fully even if the view is partly scrolled away (invisible).

Returns:
The current border. May be `null`.

See Also:
`getInnerBorder()`

---

setOuterBorder

`public void setOuterBorder(javax.swing.border.Border border)`

Property: The outer border around the header. The border will be installed on the scroll pane so that it will normally always be shown fully even if the view is partly scrolled away (invisible).

Parameters:
`border` - The new border. May be `null`.

See Also:
`setInnerBorder(Border)`

(continued from last page)

getHeaderRows

`public CellDecorationRow[] getHeaderRows()`

Property: An array of rows that makes out the different rows or colums for the header (rows for north/south headers and colums for east/west headers). One row normally represents a date range type, such as weeks.

If the date area shows day cells the cells in the header can be merged. This will look like seven cells in the header will look like one.

Example for three rows: Month, week and days:

```
-------------------------------------------------------------------
|              January           |            February          |
-------------------------------------------------------------------
|   w01  |  w02  |  w03  |  w04  |  w05  |  w06   |  w07   |w08 |
-------------------------------------------------------------------
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
-------------------------------------------------------------------
<pre>
```

Returns:

The current rows. Never `null`.

---

setHeaderRows

`public void setHeaderRows(CellDecorationRow[] rows)`

Property: An array of rows that makes out the different rows or colums for the header (rows for north/south headers and colums for east/west headers). One row normally represents a date range type, such as weeks.

If the date area shows day cells the cells in the header can be merged. This will look like seven cells in the header will look like one.

Example for three rows: Month, week and days:

```
-------------------------------------------------------------------
|              January           |            February          |
-------------------------------------------------------------------
|   w01  |  w02  |  w03  |  w04  |  w05  |  w06   |  w07   |w08 |
-------------------------------------------------------------------
| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
-------------------------------------------------------------------
<pre>
```

Parameters:

`rows` - The new rows. Should noramlly be at least one element and thus not `null`.

getMouseEffectsEnabled

```
public boolean getMouseEffectsEnabled()
```

Property: If mouse listeners should be installed on the header to make live effects enabled on the cell rows. This must be enabled for any effects to be visible. The actual effects must also be set on the `com.miginfocom.calendar.header.CellDecorationRows` of course.

If date selection in the header is to be listened for this must be enabled.

Returns:
If currently enabled.

See Also:
`addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)`

setMouseEffectsEnabled

```
public void setMouseEffectsEnabled(boolean b)
```

Property: If mouse listeners should be installed on the header to make live effects enabled on the cell rows. This must be enabled for any effects to be visible. The actual effects must also be set on the `com.miginfocom.calendar.header.CellDecorationRows` of course.

If date selection in the header is to be listened for this must be enabled.

Parameters:
`b` - If enabled.

See Also:
`addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)`

addDateChangeListener

```
public void addDateChangeListener(DateChangeListener l)
```

This call is transmitted to the `addDateChangeListener(com.miginfocom.util.dates.DateChangeListener)`

Note that mouse effects must be enabled. See `setMouseEffectsEnabled(boolean)`

See Also:
`addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)`

addDateChangeListener

```
public void addDateChangeListener(DateChangeListener l,
          boolean asWeakRef)
```

This call is transmitted to the `addDateChangeListener(com.miginfocom.util.dates.DateChangeListener, boolean)`

Note that mouse effects must be enabled. See `setMouseEffectsEnabled(boolean)`

removeDateChangeListener

```
public void removeDateChangeListener(DateChangeListener l)
```

This call is transmitted to the `removeDateChangeListener(com.miginfocom.util.dates.DateChangeListener)`

com.miginfocom.beans
Class DatePickerBean

```
java.lang.Object
   |
   +-DatePicker
      |
      +-com.miginfocom.beans.DatePickerBean
```

public class **DatePickerBean**
extends DatePicker

This is a visual JavaBean version of `com.miginfocom.calendar.DatePicker`.

Constructor Summary

public	DatePickerBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

DatePickerBean

`public` **`DatePickerBean`**`()`

com.miginfocom.beans
Class DateSpinnerBean

```
java.lang.Object
   |-SlimDateSpinner
      |-com.miginfocom.beans.DateSpinnerBean
```

public class **DateSpinnerBean**
extends SlimDateSpinner

A JavaBean component that shows a date field and arrows.

Constructor Summary

public	DateSpinnerBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

DateSpinnerBean

`public` **`DateSpinnerBean`**`()`

com.miginfocom.beans
Class DemoDataBean

```
java.lang.Object
   ┗-Category
      ┗-com.miginfocom.beans.DemoDataBean
```

All Implemented Interfaces:
Serializable

---

public class **DemoDataBean**
extends Category
implements Serializable

A JavaBean that can be used to create some random demo activities.

This class should only be used to test activity handling. Performance is sub-par for anything else since the demo data will be recreated for every setXxx method called.

The algorithm start from the earliest time (set in the propertis of this class) and the randomizing values (normally between some min/max value) for all properties of the activity until the end time is reached.

See Also:
`createActivities(com.miginfocom.util.dates.DateRangeI, String[], String[], Object[], Object[][], int, int, int, int, int, int)`

Constructor Summary

public	`DemoDataBean()`

Method Summary

void	`createDemoData()` This method will create the demo data.
ImmutableDateRange	`geDataDateRange()` Property: The date range that the demo data will be created for.
String	`getActivityCategories()` Property: A comma separated list of main categories that all activities will get one (random) of.
String	`getActivityDepositoryContext()` Property: The context used to get the depository to put all generated demo activities in.
String	`getCategories()` Property: A comma separated list with special chars of main categories that will be created in the `com.miginfocom.calendar.category.CategoryDepository`.
String	`getDescriptionTexts()` Property: A comma separated list of description texts that will randomly be selected from for every created activity.
int	`getGapMinutesMax()` Property: The maximum minutes that should pass between two generated activities.

int	getGapMinutesMin() Property: The minimum minutes that has to pass between two generated activities.
int	getLengthMinutesMax() Property: The maximum number of minutes that an activity should span.
int	getLengthMinutesMin() Property: The minimum number of minutes that an activity should span.
Locale	getLocale() Property: The locale to set for every created com.miginfocom.calendar.activity.Activity.
int	getMaxActivityCount() Property: The maximum number of activities generated.
int	getRoundToMinutes() Property: The number of minutes to round to.
String	getSummaryTexts() Property: A comma separated list of summaries that will randomly be selected from for every created activity.
TimeZone	getTimeZone() Property: The time zone to set for every created com.miginfocom.calendar.activity.Activity.
boolean	isDemoDataCreated() Returns if this bean should generate demo data or not.
boolean	isEnabled() Property: If the demo data bean is enabled or not.
boolean	isOnlyDesignTime() Property: If the demo date should only be created in a design time environment.
void	removeDemoData() This method will remove the demo data.
void	setActivityCategories(String actCats) Property: A comma separated list of main categories that all activities will get one (random) of.
void	setActivityDepositoryContext(String ctx) Property: The context used to get the depository to put all generated demo activities in.
void	setCategories(String cats) Property: A comma separated list with special chars of main categories that will be created in the com.miginfocom.calendar.category.CategoryDepository.
void	setDataDateRange(ImmutableDateRange dateRange) Property: The date range that the demo data will be created for.
void	setDescriptionTexts(String descrTexts) Property: A comma separated list of description texts that will randomly be selected from for every created activity.
void	setEnabled(boolean b) Property: If the demo data bean is enabled or not.

void	`setGapMinutesMax(int minutes)` Property: The maximum minutes that should pass between two generated activities.
void	`setGapMinutesMin(int minutes)` Property: The minimum minutes that has to pass between two generated activities.
void	`setLengthMinutesMax(int minutes)` Property: The maximum number of minutes that an activity should span.
void	`setLengthMinutesMin(int minutes)` Property: The minimum number of minutes that an activity should span.
void	`setLocale(Locale locale)` Property: The locale to set for every created `com.miginfocom.calendar.activity.Activity`.
void	`setMaxActivityCount(int count)` Property: The maximum number of activities generated.
void	`setOnlyDesignTime(boolean b)` Property: If the demo date should only be created in a design time environment.
void	`setRoundToMinutes(int minutes)` Property: The number of minutes to round to.
void	`setSummaryTexts(String sumTexts)` Property: A comma separated list of summaries that will randomly be selected from for every created activity.
void	`setTimeZone(TimeZone timeZone)` Property: The time zone to set for every created `com.miginfocom.calendar.activity.Activity`.

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

DemoDataBean

`public` **`DemoDataBean`**`()`

Methods

isEnabled

`public boolean` **`isEnabled`**`()`

Property: If the demo data bean is enabled or not. If not enabled the demo data will never be created.

Returns:

If the demo data bean is enabled or not.

(continued from last page)

setEnabled

`public void setEnabled(boolean b)`

Property: If the demo data bean is enabled or not. If not enabled the demo data will never be created.

Parameters:
`b` - If the demo data bean should be enabled or not.

---

isDemoDataCreated

`public boolean isDemoDataCreated()`

Returns if this bean should generate demo data or not.

Returns:
`true` is demo data has been generated.

---

createDemoData

`public void createDemoData()`

This method will create the demo data.

See Also:
`getInstance(String)`
`setEnabled(boolean)`

---

removeDemoData

`public void removeDemoData()`

This method will remove the demo data.

See Also:
`getInstance(String)`

---

getActivityDepositoryContext

`public String getActivityDepositoryContext()`

Property: The context used to get the depository to put all generated demo activities in.

Returns:
The current context. May be `null`.

See Also:
`getInstance(String)`

---

setActivityDepositoryContext

`public void setActivityDepositoryContext(String ctx)`

Property: The context used to get the depository to put all generated demo activities in.

This method will recreated the demo data which may take some time.

Parameters:
`ctx` - The new context. May be `null`.

(continued from last page)

See Also:
`getInstance(String)`
`createActivities(com.miginfocom.util.dates.DateRangeI, String[], String[], Object[], Object[][], int, int, int, int, int, int)`

---

getCategories

```
public String getCategories()
```

Property: A comma separated list with special chars of main categories that will be created in the `com.miginfocom.calendar.category.CategoryDepository`.

Returns:
The current comma separeted list with categories (and special chars).

See Also:
`setCategories(String)`

---

setCategories

```
public void setCategories(String cats)
```

Property: A comma separated list with special chars of main categories that will be created in the `com.miginfocom.calendar.category.CategoryDepository`.

The list can be used to create a hierarchy (tree) of categories.

This method will recreated the demo data which may take some time.

Parameters:
`cats` - The comma separated category ids. For every element, if first char is a "+" the category `catIDs[catIx]` will be a folder beneath `addToNode` and the rest will be added to this new fonder. If the first char is one or more "-" the category `catIDs[catIx]` will be added to the parent that many levels up from `addToNode`. If not "+" or "-" then `catIDs[catIx]` will just be added to `addToNode`.

"-" and "+" can be combined but "-" must always be first if so. For instacne "-+Folder" creates a folder in the parent category.

---

getActivityCategories

```
public String getActivityCategories()
```

Property: A comma separated list of main categories that all activities will get one (random) of.

Returns:
The current comma separated list. May be `null`.

See Also:
`createActivities(com.miginfocom.util.dates.DateRangeI, String[], String[], Object[], Object[][], int, int, int, int, int, int)`

---

setActivityCategories

```
public void setActivityCategories(String actCats)
```

Property: A comma separated list of main categories that all activities will get one (random) of.

This method will recreated the demo data which may take some time.

Parameters:
`actCats` - The new comma separated list. May be `null` in which case the activities will get no categories.

(continued from last page)

See Also:
`createActivities(com.miginfocom.util.dates.DateRangeI, String[], String[], Object[], Object[][], int, int, int, int, int, int)`

---

getSummaryTexts

`public String getSummaryTexts()`

Property: A comma separated list of summaries that will randomly be selected from for every created activity.

Returns:
The current list. Never `null` but may be "":

---

setSummaryTexts

`public void setSummaryTexts(String sumTexts)`

Property: A comma separated list of summaries that will randomly be selected from for every created activity.

This method will recreated the demo data which may take some time.

Parameters:
`sumTexts` - The new list. `null` is converted to "":

---

getDescriptionTexts

`public String getDescriptionTexts()`

Property: A comma separated list of description texts that will randomly be selected from for every created activity.

Returns:
The current list. Never `null` but may be "":

---

setDescriptionTexts

`public void setDescriptionTexts(String descrTexts)`

Property: A comma separated list of description texts that will randomly be selected from for every created activity.

This method will recreated the demo data which may take some time.

Parameters:
`descrTexts` - The new list. `null` is converted to "":

---

getTimeZone

`public TimeZone getTimeZone()`

Property: The time zone to set for every created `com.miginfocom.calendar.activity.Activity`.

Returns:
The current time zone used. May be `null`.

---

setTimeZone

`public void setTimeZone(TimeZone timeZone)`

(continued from last page)

Property: The time zone to set for every created `com.miginfocom.calendar.activity.Activity`.

This method will recreated the demo data which may take some time.

Parameters:
`timeZone` - The new time zone to use. May be `null`.

---

getLocale

`public Locale getLocale()`

Property: The locale to set for every created `com.miginfocom.calendar.activity.Activity`.

Returns:
The current locale used. May be `null`.

---

setLocale

`public void setLocale(Locale locale)`

Property: The locale to set for every created `com.miginfocom.calendar.activity.Activity`.

This method will recreated the demo data which may take some time.

Parameters:
`locale` - The new time zone to use. May be `null`.

---

geDataDateRange

`public ImmutableDateRange geDataDateRange()`

Property: The date range that the demo data will be created for. Not `null`.

Returns:
The current range. Not `null`.

---

setDataDateRange

`public void setDataDateRange(ImmutableDateRange dateRange)`

Property: The date range that the demo data will be created for. Not `null`.

The demo data will not be recreated. Call `recreateDemoData()` for that.

Parameters:
`dateRange` - The new range. Not `null`.

---

getGapMinutesMin

`public int getGapMinutesMin()`

Property: The minimum minutes that has to pass between two generated activities. If a negative value they may sometimes overlap.

Returns:
The current number of minutes. May be negative.

(continued from last page)

setGapMinutesMin

```
public void setGapMinutesMin(int minutes)
```

Property: The minimum minutes that has to pass between two generated activities. If a negative value they may sometimes overlap.

This method will recreated the demo data which may take some time.

Parameters:
`minutes` - The new number of minutes. May be negative.

---

getGapMinutesMax

```
public int getGapMinutesMax()
```

Property: The maximum minutes that should pass between two generated activities. Should not be negative.

Returns:
The current number of minutes. Not negative.

---

setGapMinutesMax

```
public void setGapMinutesMax(int minutes)
```

Property: The maximum minutes that should pass between two generated activities. Should not be negative.

This method will recreated the demo data which may take some time.

Parameters:
`minutes` - The new number of minutes. Not negative.

---

getRoundToMinutes

```
public int getRoundToMinutes()
```

Property: The number of minutes to round to. For instance if `5` all activities start and end times will be rounded to five minutes intervals (0, 5, 10, 15 etc..).

Returns:
The current value.

---

setRoundToMinutes

```
public void setRoundToMinutes(int minutes)
```

Property: The number of minutes to round to. For instance if `5` all activities start and end times will be rounded to five minutes intervals (0, 5, 10, 15 etc..).

This method will recreated the demo data which may take some time.

Parameters:
`minutes` - The new value.

---

getLengthMinutesMin

```
public int getLengthMinutesMin()
```

Property: The minimum number of minutes that an activity should span.

(continued from last page)

Returns:
The current value.

setLengthMinutesMin

```
public void setLengthMinutesMin(int minutes)
```

Property: The minimum number of minutes that an activity should span.

This method will recreated the demo data which may take some time.

Parameters:
`minutes` - The new value.

getLengthMinutesMax

```
public int getLengthMinutesMax()
```

Property: The maximum number of minutes that an activity should span.

Returns:
The current value.

setLengthMinutesMax

```
public void setLengthMinutesMax(int minutes)
```

Property: The maximum number of minutes that an activity should span.

This method will recreated the demo data which may take some time.

Parameters:
`minutes` - The new value.

getMaxActivityCount

```
public int getMaxActivityCount()
```

Property: The maximum number of activities generated. This is mostly to avoid creation of too many due to choosing the the properties for this bean badly. It's a fail safe...

Returns:
The current value. Default is 200.

setMaxActivityCount

```
public void setMaxActivityCount(int count)
```

Property: The maximum number of activities generated. This is mostly to avoid creation of too many due to choosing the the properties for this bean badly. It's a fail safe...

This method will recreated the demo data which may take some time.

Parameters:
`count` - The new value. Default is 200.

isOnlyDesignTime

```
public boolean isOnlyDesignTime()
```

(continued from last page)

Property: If the demo date should only be created in a design time environment. This usually means in the designer of the IDE.

Returns:

The current value. Default is false.

---

setOnlyDesignTime

```
public void setOnlyDesignTime(boolean b)
```

Property: If the demo date should only be created in a design time environment. This usually means in the designer of the IDE.

This method will recreated the demo data which may take some time.

Parameters:

`b` - The new value.

com.miginfocom.beans
Class GridDimensionLayoutBean

```
java.lang.Object
   ┗-com.miginfocom.beans.AbstractBean
      ┗-com.miginfocom.beans.GridDimensionLayoutBean
```

All Implemented Interfaces:
Serializable

---

public class **GridDimensionLayoutBean**
extends AbstractBean

A bean that shows set the properties for how a dimension (horozontal or vertical) of a `com.miginfocom.calendar.grid.Grid` should be laied out.

It set the size specs for the rows (cells) in the dimension it describes.

It has three major types of rows:

1. **Normal** - Basically a row that isn't any of the other types.
2. **Compressed** - A row designated to be compressed by the properties `CompressRowsFormat` and `CompressRowsRanges`
3. **Sub row** - A row (only for the secondary dimension) that is a sub row of another sub row or one of the main rows. Main rows are the first level cells in the grid.
4.

Field Summary

`public static final`	`ABSOLUTE_CELLS` Value: **0**
`public static final`	`DATE` Value: **3**
`public static final`	`DAY_OF_WEEK` Value: **2**
`public static final`	`TIME_OF_DAY` Value: **1**

Constructor Summary

`public`	`GridDimensionLayoutBean()` Constructor.

Method Summary

`GridDimensionLayout`	`createLayout(Grid grid, int dimIx)` Returns the layout that this bean represents.

int	getCompressRowsFormat() Property: The type (unit) for the CompressRowsRanges property.
String	getCompressRowsRanges() Property: A string that describles what rows, or row intervals, that should be compressed.
SizeSpec	getRowSizeCompressed() Property: The minimum, maximum and preferred size for a compressed row.
SizeSpec	getRowSizeNormal() Property: The minimum, maximum and preferred size for a normal row (not compressed).
SizeSpec	getSubRowSizeExpandedFolder() Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered expanded.
SizeSpec	getSubRowSizeFoldedFolder() Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered folded.
SizeSpec	getSubRowSizeLeaf() Property: The minimum, maximum and preferred size for a sub row that has no sub rows itself (it's a leaf).
boolean	isExpandRowsToFit() Property: If the rows in the dimension that this object represents should be expanded to fit the available space.
boolean	isExpandSubRowsToFit() Property: If the **sub**-rows should be expanded to fit the available space.
void	setCompressRowsFormat(int format) Property: The type (unit) for the CompressRowsRanges property.
void	setCompressRowsRanges(String s) Property: A string that describles what rows, or row intervals, that should be compressed.
void	setExpandRowsToFit(boolean b) Property: If the rows in the dimension that this object represents should be expanded to fit the available space.
void	setExpandSubRowsToFit(boolean b) Property: If the **sub**-rows should be expanded to fit the available space.
void	setRowSizeCompressed(SizeSpec size) Property: The minimum, maximum and preferred size for a compressed row.
void	setRowSizeNormal(SizeSpec size) Property: The minimum, maximum and preferred size for a normal row (not compressed).
void	setSubRowSizeExpandedFolder(SizeSpec size) Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered expanded.
void	setSubRowSizeFoldedFolder(SizeSpec size) Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered folded.
void	setSubRowSizeLeaf(SizeSpec size) Property: The minimum, maximum and preferred size for a sub row that has no sub rows itself (it's a leaf).

Methods inherited from class `com.miginfocom.beans.AbstractBean`
`addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents`

Methods inherited from class `java.lang.Object`
`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Fields

ABSOLUTE_CELLS

```
public static final int ABSOLUTE_CELLS
```

Constant value: **0**

TIME_OF_DAY

```
public static final int TIME_OF_DAY
```

Constant value: **1**

DAY_OF_WEEK

```
public static final int DAY_OF_WEEK
```

Constant value: **2**

DATE

```
public static final int DATE
```

Constant value: **3**

Constructors

GridDimensionLayoutBean

```
public GridDimensionLayoutBean()
```

Constructor.

Methods

createLayout

```
public GridDimensionLayout createLayout(Grid grid,
         int dimIx)
```

Returns the layout that this bean represents.

(continued from last page)

Returns:
The layout that this bean represents.

isExpandRowsToFit

`public boolean isExpandRowsToFit()`

Property: If the rows in the dimension that this object represents should be expanded to fit the available space. Note that this is only used if the size properties for the rows themeselves doesn't specify anything else.

Returns:
The current value, `true` is default.

setExpandRowsToFit

`public void setExpandRowsToFit(boolean b)`

Property: If the rows in the dimension that this object represents should be expanded to fit the available space. Note that this is only used if the size properties for the rows themeselves doesn't specify anything else.

Parameters:
`b` - The new value, `true` is default.

isExpandSubRowsToFit

`public boolean isExpandSubRowsToFit()`

Property: If the **sub**-rows should be expanded to fit the available space. Note that this is only used if the size properties for the rows themeselves doesn't specify anything else. Also note that this value is only interesting if this bean represents the secondary dimension, since only the secondary dimension has sub rows.

Returns:
The current value, `true` is default.

setExpandSubRowsToFit

`public void setExpandSubRowsToFit(boolean b)`

Property: If the **sub**-rows should be expanded to fit the available space. Note that this is only used if the size properties for the rows themeselves doesn't specify anything else. Also note that this value is only interesting if this bean represents the secondary dimension, since only the secondary dimension has sub rows.

Parameters:
`b` - The new value, `true` is default.

getCompressRowsRanges

`public String getCompressRowsRanges()`

Property: A string that describles what rows, or row intervals, that should be compressed. The unit used can be set with `setCompressRowsFormat(int)`.

The string is comma separated invervalls using english locale/words/formatting. For example:

- "00.00-06.00,18.00-24.00" (time of day)
- "0-10,14-17,30-40,42,44" (absolute cells)
- "mon-wed,sat-sun" (week days)
- "20041231-20050131,20060231-2007231" (dates, yyyyMMdd)

(continued from last page)

Returns:
The current range(s). Not `null` but can be empty.

setCompressRowsRanges

`public void setCompressRowsRanges(String s)`

Property: A string that describles what rows, or row intervals, that should be compressed. The unit used can be set with `setCompressRowsFormat(int)`.

The string is comma separated invervalls using english locale/words/formatting. For example:

- "00.00-06.00,18.00-24.00" (time of day)
- "0-10,14-17,30-40,42,44" (absolute cells)
- "mon-wed,sat-sun" (week days)
- "20041231-20050131,20060231-2007231" (dates, yyyyMMdd)

Parameters:
`s` - The new range(s). Not `null` but can be empty.

getCompressRowsFormat

`public int getCompressRowsFormat()`

Property: The type (unit) for the `CompressRowsRanges` property.

Valid formats are:

- `ABSOLUTE_CELLS` - The cell numbers starting from 0.
- `TIME_OF_DAY` - Time of day of the form: "HH.mm"
- `DAY_OF_WEEK` - The day of week. It's enough to use the first two letters. E.g "mo" or "tue".
- `DATE` - The date without any time on the form: "yyyyMMdd"

Returns:
The current format. `ABSOLUTE_CELLS` is default.

setCompressRowsFormat

`public void setCompressRowsFormat(int format)`

Property: The type (unit) for the `CompressRowsRanges` property.

Valid formats are:

- `ABSOLUTE_CELLS` - The cell numbers starting from 0.
- `TIME_OF_DAY` - Time of day of the form: "HH.mm"
- `DAY_OF_WEEK` - The day of week. It's enough to use the first two letters. E.g "mo" or "tue".
- `DATE` - The date without any time on the form: "yyyyMMdd"

Parameters:
`format` - The new format. `ABSOLUTE_CELLS` is default.

(continued from last page)

getRowSizeNormal

`public SizeSpec getRowSizeNormal()`

Property: The minimum, maximum and preferred size for a normal row (not compressed).

Returns:
The current size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`.

See Also:
`SizeSpec`

---

setRowSizeNormal

`public void setRowSizeNormal(SizeSpec size)`

Property: The minimum, maximum and preferred size for a normal row (not compressed).

Parameters:
`size` - The new size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`. The value will be stored and not cloned (it's final so this shouldn't matter though).

See Also:
`SizeSpec`

---

getRowSizeCompressed

`public SizeSpec getRowSizeCompressed()`

Property: The minimum, maximum and preferred size for a compressed row. Which rows that are compressed are specified by the properties `CompressRowsFormat` and `CompressRowsRanges`.

Returns:
The current size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`.

See Also:
`SizeSpec`
`setCompressRowsFormat(int)`
`setCompressRowsRanges(String)`

---

setRowSizeCompressed

`public void setRowSizeCompressed(SizeSpec size)`

Property: The minimum, maximum and preferred size for a compressed row. Which rows that are compressed are specified by the properties `CompressRowsFormat` and `CompressRowsRanges`.

Parameters:
`size` - The new size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`. The value will be stored and not cloned (it's final so this shouldn't matter though).

See Also:
`SizeSpec`
`setCompressRowsFormat(int)`
`setCompressRowsRanges(String)`

---

getSubRowSizeLeaf

`public SizeSpec getSubRowSizeLeaf()`

(continued from last page)

Property: The minimum, maximum and preferred size for a sub row that has no sub rows itself (it's a leaf). This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Returns:

The current size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`.

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

---

setSubRowSizeLeaf

`public void` **`setSubRowSizeLeaf`**`(SizeSpec size)`

Property: The minimum, maximum and preferred size for a sub row that has no sub rows itself (it's a leaf). This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Parameters:

`size` - The new size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`. The value will be stored and not cloned (it's final so this shouldn't matter though).

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

---

getSubRowSizeExpandedFolder

`public SizeSpec` **`getSubRowSizeExpandedFolder`**`()`

Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered expanded. This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Returns:

The current size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`.

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

---

setSubRowSizeExpandedFolder

`public void` **`setSubRowSizeExpandedFolder`**`(SizeSpec size)`

Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered expanded. This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Parameters:

`size` - The new size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`. The value will be stored and not cloned (it's final so this shouldn't matter though).

(continued from last page)

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

---

getSubRowSizeFoldedFolder

`public SizeSpec` **`getSubRowSizeFoldedFolder`**`()`

Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered folded. This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Returns:

The current size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`.

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

---

setSubRowSizeFoldedFolder

`public void` **`setSubRowSizeFoldedFolder`**`(SizeSpec size)`

Property: The minimum, maximum and preferred size for a sub row that **has** sub rows itself and the row is considered folded. This is only interesting (hence used) if this bean denotes the layout for the secondary dimension since only the secondary dimension can have sub rows.

Parameters:

`size` - The new size spec. Not `null` but the minimum, maximum and preferred size in the spec can all be `null`. The value will be stored and not cloned (it's final so this shouldn't matter though).

See Also:

`SizeSpec`
`setSubRowSizeExpandedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeFoldedFolder(com.miginfocom.util.gfx.geometry.SizeSpec)`
`setSubRowSizeLeaf(com.miginfocom.util.gfx.geometry.SizeSpec)`

com.miginfocom.beans
Class NorthCategoryHeaderBean

```
java.lang.Object
   ┗-com.miginfocom.beans.AbstractBean
      ┗-com.miginfocom.beans.AbstractHeaderBean
         ┗-com.miginfocom.beans.CategoryHeaderBean
            ┗-com.miginfocom.beans.NorthCategoryHeaderBean
```

All Implemented Interfaces:
Serializable

---

public class **NorthCategoryHeaderBean**
extends CategoryHeaderBean

Exactly same as a `CategoryHeaderBean` but with a different set of default optimized to get started with a top header faster.

`CategoryHeaderBean` can be configured to work as both top or left headers, but the properties to get it to look good for either is quite different.

Constructor Summary

public	NorthCategoryHeaderBean()

Methods inherited from class com.miginfocom.beans.CategoryHeaderBean

addInteractionListener, addInteractionListener, getCategoryDepth, getCellBorder, getCellCursor, getForcedHeaderSize, getHeader, getHeaderLevels, getKnobCursor, getKnobExpandedImage, getKnobFoldedImage, getKnobFoldOnPress, getKnobImageAlignX, getKnobImageAlignY, getLabelCursor, getLabelFoldOnPress, getNoExpandedFolderGridLine, getRowFolderImage, getRowImageAlignX, getRowImageAlignY, getRowLeafImage, getTextAntiAlias, interactionOccured, isFolderOverlapChildren, removeInteractionListener, setCellBorder, setCellCursor, setDateAreaContainer, setFolderOverlapChildren, setForcedHeaderSize, setHeaderLevels, setKnobCursor, setKnobExpandedImage, setKnobFoldedImage, setKnobFoldOnPress, setKnobImageAlignX, setKnobImageAlignY, setLabelCursor, setLabelFoldOnPress, setNoExpandedFolderGridLine, setRowFolderImage, setRowImageAlignX, setRowImageAlignY, setRowLeafImage, setTextAntiAlias

Methods inherited from class com.miginfocom.beans.AbstractHeaderBean

getBackgroundPaint, getContainer, getEdge, getExpandToCorner, isVisible, revalidateRepaintContainer, setBackgroundPaint, setDateAreaContainer, setEdge, setExpandToCorner, setVisible

Methods inherited from class com.miginfocom.beans.AbstractBean

addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

NorthCategoryHeaderBean

public **NorthCategoryHeaderBean**()

com.miginfocom.beans
Class PaintPanelBean

```
java.lang.Object
   |-BackgroundPanel
      |-com.miginfocom.beans.PaintPanelBean
```

public class **PaintPanelBean**
extends BackgroundPanel

A normal `javax.swing.JPanel` that can take a generic `java.awt.Paint` object and use that as the background.

Constructor Summary

public	PaintPanelBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

PaintPanelBean

`public` **`PaintPanelBean`**`()`

com.miginfocom.beans
Class PrintPreviewBean

```
java.lang.Object
   |-java.awt.Component
      |-java.awt.Container
         |-javax.swing.JComponent
            |-javax.swing.JPanel
               |-com.miginfocom.beans.PrintPreviewBean
```

All Implemented Interfaces:
Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.swing.TransferHandler.HasGetTransferHandler, Serializable, javax.accessibility.Accessible

---

public final class **PrintPreviewBean**
extends javax.swing.JPanel

Fields inherited from class `javax.swing.JComponent`
`TOOL_TIP_TEXT_KEY`, `UNDEFINED_CONDITION`, `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`

Fields inherited from class `java.awt.Component`
`BOTTOM_ALIGNMENT`, `CENTER_ALIGNMENT`, `LEFT_ALIGNMENT`, `RIGHT_ALIGNMENT`, `TOP_ALIGNMENT`

Fields inherited from interface `java.awt.image.ImageObserver`
`ABORT`, `ALLBITS`, `ERROR`, `FRAMEBITS`, `HEIGHT`, `PROPERTIES`, `SOMEBITS`, `WIDTH`

Constructor Summary

public	PrintPreviewBean()

Method Summary

void	doLayout()
DateAreaContainer	getDateAreaContainer() Property: The `com.miginfocom.calendar.datearea.DateAreaContainer` that is to be previewed.
boolean	getFlowPages() Property: If set to `true` the pages will will flow like `java.awt.FlowLayout` and while maintaining the same aspect ratio flow the pages from left to right and then top to bottom.
float	getLayoutAlignmentX() Property: How to align the preview panels as a group if there are free space to distribute.

float	getLayoutAlignmentY() Property: How to align the preview panels as a group if there are free space to distribute.
java.awt.Dimension	getMaximumSize()
java.awt.Dimension	getMinimumSize()
java.awt.print.PageFormat	getPageFormat() Property: The java.awt.print.PageFormat that will be used in the print preview.
int	getPageGapX() Property: The number of horizontal pixels between two adjacent pages.
int	getPageGapY() Property: The number of vertical pixels between two adjacent pages.
java.awt.Dimension	getPreferredSize()
javax.swing.border.Border	getPreviewPanelsBorder() Property: The border that will be set for every preview panel.
PrintSpecification	getPrintSpecification() Property: The specifiaction that outlines how this date area container should be printed.
double	getScreenDpi() Property: The dots (pixels) per inch that the screen physically have.
double	getZoomMaximum() Property: The maximum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.
double	getZoomMinimum() Property: The minimum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.
double	getZoomPreferred() Property: The prererred size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.
void	invalidate()
void	setDateAreaContainer(DateAreaContainer container) Property: The com.miginfocom.calendar.datearea.DateAreaContainer that is to be previewed.
void	setFlowPages(boolean b) Property: If set to true the pages will will flow like java.awt.FlowLayout and while maintaining the same aspect ratio flow the pages from left to right and then top to bottom.
void	setLayout(java.awt.LayoutManager lm)
void	setLayoutAlignmentX(float alignX) Property: How to align the preview panels as a group if there are free space to distribute.
void	setLayoutAlignmentY(float alignY) Property: How to align the preview panels as a group if there are free space to distribute.

void	setPageFormat(java.awt.print.PageFormat pf) Property: The java.awt.print.PageFormat that will be used in the print preview.
void	setPageGapX(int gap) Property: The number of horizontal pixels between two adjacent pages.
void	setPageGapY(int gap) Property: The number of vertical pixels between two adjacent pages.
void	setPreviewPanelsBorder(javax.swing.border.Border border) Property: The border that will be set for every preview panel.
void	setPrintSpecification(PrintSpecification printSpec) Property: The specifiaction that outlines how this date area container should be printed.
void	setScreenDpi(double dpi) Property: The dots (pixels) per inch that the screen physically have.
void	setZoomMaximum(double zoom) Property: The maximum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.
void	setZoomMinimum(double zoom) Property: The minimum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.
void	setZoomPreferred(double zoom) Property: The prererred size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%.

Methods inherited from class javax.swing.JPanel

getAccessibleContext, getUI, getUIClassID, setUI, updateUI

Methods inherited from class javax.swing.JComponent

addAncestorListener, addNotify, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, getAccessibleContext, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getMaximumSize, getMinimumSize, getNextFocusableComponent, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getUIClassID, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getWidth, getVisibleRect, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintImmediately, paintImmediately, print, printAll, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update, updateUI

Methods inherited from class `java.awt.Container`

add, add, add, add, add, addContainerListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getAlignmentX, getAlignmentY, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getInsets, getLayout, getListeners, getMaximumSize, getMinimumSize, getMousePosition, getPreferredSize, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paint, paintComponents, preferredSize, print, printComponents, remove, remove, removeAll, removeContainerListener, removeNotify, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, setLayout, transferFocusBackward, transferFocusDownCycle, update, validate

Methods inherited from class `java.awt.Component`

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, addNotify, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, bounds, checkImage, checkImage, contains, contains, createImage, createImage, createVolatileImage, createVolatileImage, deliverEvent, disable, dispatchEvent, doLayout, enable, enable, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getAccessibleContext, getAlignmentX, getAlignmentY, getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds, getColorModel, getComponentAt, getComponentAt, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeys, getFocusTraversalKeysEnabled, getFont, getFontMetrics, getForeground, getGraphics, getGraphicsConfiguration, getHeight, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getListeners, getLocale, getLocation, getLocation, getLocationOnScreen, getMaximumSize, getMinimumSize, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPreferredSize, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getSize, getToolkit, getTreeLock, getWidth, getX, getY, gotFocus, handleEvent, hasFocus, hide, imageUpdate, inside, invalidate, isBackgroundSet, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled, isFocusable, isFocusCycleRoot, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isOpaque, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, layout, list, list, list, list, list, locate, location, lostFocus, minimumSize, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paint, paintAll, postEvent, preferredSize, prepareImage, prepareImage, print, printAll, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, requestFocus, requestFocusInWindow, reshape, resize, resize, setBackground, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeys, setFocusTraversalKeysEnabled, setFont, setForeground, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setSize, setSize, setVisible, show, show, size, toString, transferFocus, transferFocusBackward, transferFocusUpCycle, update, validate

Methods inherited from class `java.lang.Object`

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Methods inherited from interface `java.awt.image.ImageObserver`

imageUpdate

Methods inherited from interface `java.awt.MenuContainer`

getFont, postEvent, remove

Methods inherited from interface `javax.swing.TransferHandler.HasGetTransferHandler`

getTransferHandler

Methods inherited from interface `javax.accessibility.Accessible`

getAccessibleContext

Constructors

PrintPreviewBean

public **PrintPreviewBean**()

Methods

invalidate

public void **invalidate**()

setLayout

public void **setLayout**(java.awt.LayoutManager lm)

getPreferredSize

public java.awt.Dimension **getPreferredSize**()

getMaximumSize

public java.awt.Dimension **getMaximumSize**()

getMinimumSize

public java.awt.Dimension **getMinimumSize**()

doLayout

public void **doLayout**()

getDateAreaContainer

public DateAreaContainer **getDateAreaContainer**()

Property: The com.miginfocom.calendar.datearea.DateAreaContainer that is to be previewed.

Returns:

(continued from last page)

The container. May be `null`.

setDateAreaContainer

`public void **setDateAreaContainer**(DateAreaContainer container)`

Property: The `com.miginfocom.calendar.datearea.DateAreaContainer` that is to be previewed.

Parameters:

`container` - The container. May be `null` to reset. Will be saved in a weak reference so there are no memory leaks. This means that the container need to be held in a normal stong reference or the preview bean will not work.

getPrintSpecification

`public PrintSpecification **getPrintSpecification**()`

Property: The specifiaction that outlines how this date area container should be printed.

Returns:

The current specification. Not `null`.

Since:

6.0

setPrintSpecification

`public void **setPrintSpecification**(PrintSpecification printSpec)`

Property: The specifiaction that outlines how this date area container should be printed.

Parameters:

`printSpec` - The new specification. If `null` a specification will be retrieved from the `com.miginfocom.calendar.datearea.DateAreaContainer`.

Since:

6.0

getZoomMinimum

`public double **getZoomMinimum**()`

Property: The minimum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

Returns:

The current zoom level.

setZoomMinimum

`public void **setZoomMinimum**(double zoom)`

Property: The minimum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

(continued from last page)

Parameters:

`zoom` - The new zoom level.

getZoomPreferred

`public double` **`getZoomPreferred`**`()`

Property: The prererred size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

Returns:

The current zoom level.

setZoomPreferred

`public void` **`setZoomPreferred`**`(double zoom)`

Property: The prererred size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

Parameters:

`zoom` - The new zoom level.

getZoomMaximum

`public double` **`getZoomMaximum`**`()`

Property: The maximum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

Returns:

The current zoom level.

setZoomMaximum

`public void` **`setZoomMaximum`**`(double zoom)`

Property: The maximum size of and individual page expressed in percent. 1.0 is 100% and 0.0 is 0%. Values above 100% (1.0) is allowed.

100% means the size that the page will get on screen is equal to the size of the paper, so for instance if you meassure the width on the screen with a ruler a paper with the "Letter" size would be 8.5 inces wide.

Parameters:

`zoom` - The new zoom level.

getScreenDpi

`public double` **`getScreenDpi`**`()`

(continued from last page)

Property: The dots (pixels) per inch that the screen physically have. This value is used to caclulate how long a certain distance (e.g. one inch) is on the screen. Since a few screens have higher DPI than 72 and Java has no standard way to get this value you can set it here so that a page that is a certain width will get that width on the screen as well (when the zoom is 100%).

Returns:

The current value. 96 is default.

---

setScreenDpi

```
public void setScreenDpi(double dpi)
```

Property: The dots (pixels) per inch that the screen physically have. This value is used to caclulate how long a certain distance (e.g. one inch) is on the screen. Since a few screens have higher DPI than 72 and Java has no standard way to get this value you can set it here so that a page that is a certain width will get that width on the screen as well (when the zoom is 100%).

Parameters:

`dpi` - The new value. 96 is default.

---

getFlowPages

```
public boolean getFlowPages()
```

Property: If set to `true` the pages will will flow like `java.awt.FlowLayout` and while maintaining the same aspect ratio flow the pages from left to right and then top to bottom.

If `false` the layout manager will arrange the pages in the way that they look, so that the painted picture will be correct. This will mean in a x by y grid.

Returns:

The current value.

---

setFlowPages

```
public void setFlowPages(boolean b)
```

Property: If set to `true` the pages will will flow like `java.awt.FlowLayout` and while maintaining the same aspect ratio flow the pages from left to right and then top to bottom.

If `false` the layout manager will arrange the pages in the way that they look, so that the painted picture will be correct. This will mean in a x by y grid.

Parameters:

`b` - the new value.

---

setLayoutAlignmentX

```
public void setLayoutAlignmentX(float alignX)
```

Property: How to align the preview panels as a group if there are free space to distribute.

Parameters:

`alignX` - The new alignment. 0.0 to 1.0.

---

getLayoutAlignmentX

```
public float getLayoutAlignmentX()
```

Property: How to align the preview panels as a group if there are free space to distribute.

(continued from last page)

Returns:
The alignment. 0.0 to 1.0.

setLayoutAlignmentY

```
public void setLayoutAlignmentY(float alignY)
```

Property: How to align the preview panels as a group if there are free space to distribute.

Parameters:
`alignY` - The new alignment. 0.0 to 1.0.

getLayoutAlignmentY

```
public float getLayoutAlignmentY()
```

Property: How to align the preview panels as a group if there are free space to distribute.

Returns:
The alignment. 0.0 to 1.0.

getPreviewPanelsBorder

```
public javax.swing.border.Border getPreviewPanelsBorder()
```

Property: The border that will be set for every preview panel.

Returns:
The current border. May be `null`.

setPreviewPanelsBorder

```
public void setPreviewPanelsBorder(javax.swing.border.Border border)
```

Property: The border that will be set for every preview panel.

Parameters:
`border` - The new border. May be `null`.

getPageFormat

```
public java.awt.print.PageFormat getPageFormat()
```

Property: The `java.awt.print.PageFormat` that will be used in the print preview.

Returns:
The current value. Never `null`.

setPageFormat

```
public void setPageFormat(java.awt.print.PageFormat pf)
```

Property: The `java.awt.print.PageFormat` that will be used in the print preview.

Parameters:
`pf` - The new value. Never `null`.

(continued from last page)

getPageGapY

```
public int getPageGapY()
```

Property: The number of vertical pixels between two adjacent pages.

Returns:
The current gap.

---

setPageGapY

```
public void setPageGapY(int gap)
```

Property: The number of vertical pixels between two adjacent pages.

Parameters:
`gap` - The new gap.

---

getPageGapX

```
public int getPageGapX()
```

Property: The number of horizontal pixels between two adjacent pages.

Returns:
The current gap.

---

setPageGapX

```
public void setPageGapX(int gap)
```

Property: The number of horizontal pixels between two adjacent pages.

Parameters:
`gap` - The new gap.

com.miginfocom.beans
Class PrintSpecificationBean

```
java.lang.Object
   |-PrintSpecification
      |-com.miginfocom.beans.PrintSpecificationBean
```

public class **PrintSpecificationBean**
extends PrintSpecification

This is a visual JavaBean version of `com.miginfocom.util.print.PrintSpecification`.

Constructor Summary

public	PrintSpecificationBean()

Methods inherited from class `java.lang.Object`

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructors

PrintSpecificationBean

`public` **`PrintSpecificationBean`**`()`

com.miginfocom.beans
Class WestCategoryHeaderBean

```
java.lang.Object
   |
   +-com.miginfocom.beans.AbstractBean
      |
      +-com.miginfocom.beans.AbstractHeaderBean
         |
         +-com.miginfocom.beans.CategoryHeaderBean
            |
            +-com.miginfocom.beans.WestCategoryHeaderBean
```

All Implemented Interfaces:
Serializable

---

public class **WestCategoryHeaderBean**
extends CategoryHeaderBean

Exactly same as a `CategoryHeaderBean` but with a different set of default optimized to get started with a west (left) header faster.

`CategoryHeaderBean` can be configured to work as both top or left headers, but the properties to get it to look good for either is quite different.

Constructor Summary

public	WestCategoryHeaderBean()

Methods inherited from class com.miginfocom.beans.CategoryHeaderBean

addInteractionListener, addInteractionListener, getCategoryDepth, getCellBorder, getCellCursor, getForcedHeaderSize, getHeader, getHeaderLevels, getKnobCursor, getKnobExpandedImage, getKnobFoldedImage, getKnobFoldOnPress, getKnobImageAlignX, getKnobImageAlignY, getLabelCursor, getLabelFoldOnPress, getNoExpandedFolderGridLine, getRowFolderImage, getRowImageAlignX, getRowImageAlignY, getRowLeafImage, getTextAntiAlias, interactionOccured, isFolderOverlapChildren, removeInteractionListener, setCellBorder, setCellCursor, setDateAreaContainer, setFolderOverlapChildren, setForcedHeaderSize, setHeaderLevels, setKnobCursor, setKnobExpandedImage, setKnobFoldedImage, setKnobFoldOnPress, setKnobImageAlignX, setKnobImageAlignY, setLabelCursor, setLabelFoldOnPress, setNoExpandedFolderGridLine, setRowFolderImage, setRowImageAlignX, setRowImageAlignY, setRowLeafImage, setTextAntiAlias

Methods inherited from class com.miginfocom.beans.AbstractHeaderBean

getBackgroundPaint, getContainer, getEdge, getExpandToCorner, isVisible, revalidateRepaintContainer, setBackgroundPaint, setDateAreaContainer, setEdge, setExpandToCorner, setVisible

Methods inherited from class com.miginfocom.beans.AbstractBean

addPropertyChangeListener, addPropertyChangeListener, firePropertyChangeEvent, removePropertyChangeListener, setIgnorePropertyChangeEvents

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructors

WestCategoryHeaderBean

public **WestCategoryHeaderBean**()

Index

A

C

D

F

G

I

L

N

P

R

S

T

V

W