javafx.scene

Class Group

java.lang.Object
    javafx.scene.Node
        javafx.scene.Parent
            javafx.scene.Group

All Implemented Interfaces:

Styleable, EventTarget

```
@DefaultProperty(value="children")
public class Group
extends Parent
```

A Group node contains an ObservableList of children that are rendered in order whenever this node is rendered.

A Group will take on the collective bounds of its children and is not directly resizable.

Any transform, effect, or state applied to a Group will be applied to all children of that group. Such transforms and effects will NOT be included in this Group's layout bounds, however if transforms and effects are set directly on children of this Group, those will be included in this Group's layout bounds.

By default, a Group will "auto-size" its managed resizable children to their preferred sizes during the layout pass to ensure that Regions and Controls are sized properly as their state changes. If an application needs to disable this auto-sizing behavior, then it should set autoSizeChildren to false and understand that if the preferred size of the children change, they will not automatically resize (so buyer beware!).

Group Example:

```
import javafx.scene.*;
import javafx.scene.paint.*;
import javafx.scene.shape.*;
import java.lang.Math;

Group g = new Group();
for (int i = 0; i < 5; i++) {
    Rectangle r = new Rectangle();
    r.setY(i * 20);
    r.setWidth(100);
    r.setHeight(10);
    r.setFill(Color.RED);
    g.getChildren().add(r);
}
```

Since:

JavaFX 2.0

Property Summary

All Methods **Instance Methods** **Concrete Methods**

Type	Property and Description
BooleanProperty	**autoSizeChildren** Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass.

Properties inherited from class javafx.scene.Parent

needsLayout

Properties inherited from class javafx.scene.Node

accessibleHelp, accessibleRoleDescription, accessibleRole, accessibleText, blendMode, boundsInLocal, boundsInParent, cacheHint, cache, clip, cursor, depthTest, disabled, disable, effectiveNodeOrientation, effect, eventDispatcher, focused, focusTraversable, hover, id, inputMethodRequests, layoutBounds, layoutX, layoutY, localToParentTransform, localToSceneTransform, managed, mouseTransparent, nodeOrientation, onContextMenuRequested, onDragDetected, onDragDone, onDragDropped, onDragEntered, onDragExited, onDragOver, onInputMethodTextChanged, onKeyPressed, onKeyReleased, onKeyTyped, onMouseClicked, onMouseDragEntered, onMouseDragExited, onMouseDragged, onMouseDragOver, onMouseDragReleased, onMouseEntered, onMouseExited, onMouseMoved, onMousePressed, onMouseReleased, onRotate, onRotationFinished, onRotationStarted, onScrollFinished, onScroll, onScrollStarted, onSwipeDown, onSwipeLeft, onSwipeRight, onSwipeUp, onTouchMoved, onTouchPressed, onTouchReleased, onTouchStationary, onZoomFinished, onZoom, onZoomStarted, opacity, parent, pickOnBounds, pressed, rotate, rotationAxis, scaleX, scaleY, scaleZ, scene, style, translateX, translateY, translateZ, visible

Field Summary

Fields inherited from class javafx.scene.Node

BASELINE_OFFSET_SAME_AS_HEIGHT

Constructor Summary

Constructors

Constructor and Description
Group() Constructs a group.
Group(Collection<Node> children) Constructs a group consisting of the given children.
Group(Node... children) Constructs a group consisting of children.

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method and Description
BooleanProperty	**autoSizeChildrenProperty**() Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass.
ObservableList<Node>	**getChildren**() Gets the list of children of this Group.
boolean	**isAutoSizeChildren**() Gets the value of the property autoSizeChildren.
protected void	**layoutChildren**() Group implements layoutChildren such that each child is resized to its preferred size, if the child is resizable.
double	**minHeight**(double width) Returns the node's minimum height for use in layout calculations.
double	**minWidth**(double height) Returns the node's minimum width for use in layout calculations.
double	**prefHeight**(double width) Group defines the preferred height as simply being the height of its layout bounds, which in turn is simply the sum of the positions & heights of all of its children.
double	**prefWidth**(double height) Group defines the preferred width as simply being the width of its layout bounds, which in turn is simply the sum of the positions & widths of all of its children.

void	**setAutoSizeChildren**(boolean value) Sets the value of the property autoSizeChildren.

Methods inherited from class javafx.scene.Parent

computeMinHeight, computeMinWidth, computePrefHeight, computePrefWidth, getBaselineOffset, getChildrenUnmodifiable, getManagedChildren, getStylesheets, isNeedsLayout, layout, lookup, needsLayoutProperty, queryAccessibleAttribute, requestLayout, requestParentLayout, setNeedsLayout, updateBounds

Methods inherited from class javafx.scene.Node

accessibleHelpProperty, accessibleRoleDescriptionProperty, accessibleRoleProperty, accessibleTextProperty, addEventFilter, addEventHandler, applyCss, autosize, blendModeProperty, boundsInLocalProperty, boundsInParentProperty, buildEventDispatchChain, cacheHintProperty, cacheProperty, clipProperty, computeAreaInScreen, contains, contains, cursorProperty, depthTestProperty, disabledProperty, disableProperty, effectiveNodeOrientationProperty, effectProperty, eventDispatcherProperty, executeAccessibleAction, fireEvent, focusedProperty, focusTraversableProperty, getAccessibleHelp, getAccessibleRole, getAccessibleRoleDescription, getAccessibleText, getBlendMode, getBoundsInLocal, getBoundsInParent, getCacheHint, getClassCssMetaData, getClip, getContentBias, getCssMetaData, getCursor, getDepthTest, getEffect, getEffectiveNodeOrientation, getEventDispatcher, getId, getInputMethodRequests, getLayoutBounds, getLayoutX, getLayoutY, getLocalToParentTransform, getLocalToSceneTransform, getNodeOrientation, getOnContextMenuRequested, getOnDragDetected, getOnDragDone, getOnDragDropped, getOnDragEntered, getOnDragExited, getOnDragOver, getOnInputMethodTextChanged, getOnKeyPressed, getOnKeyReleased, getOnKeyTyped, getOnMouseClicked, getOnMouseDragEntered, getOnMouseDragExited, getOnMouseDragged, getOnMouseDragOver, getOnMouseDragReleased, getOnMouseEntered, getOnMouseExited, getOnMouseMoved, getOnMousePressed, getOnMouseReleased, getOnRotate, getOnRotationFinished, getOnRotationStarted, getOnScroll, getOnScrollFinished, getOnScrollStarted, getOnSwipeDown, getOnSwipeLeft, getOnSwipeRight, getOnSwipeUp, getOnTouchMoved, getOnTouchPressed, getOnTouchReleased, getOnTouchStationary, getOnZoom, getOnZoomFinished, getOnZoomStarted, getOpacity, getParent, getProperties, getPseudoClassStates, getRotate, getRotationAxis, getScaleX, getScaleY, getScaleZ, getScene, getStyle, getStyleableParent, getStyleClass, getTransforms, getTranslateX, getTranslateY, getTranslateZ, getTypeSelector, getUserData, hasProperties, hoverProperty, idProperty, inputMethodRequestsProperty, intersects, intersects, isCache, isDisable, isDisabled, isFocused, isFocusTraversable, isHover, isManaged, isMouseTransparent, isPickOnBounds, isPressed, isResizable, isVisible, layoutBoundsProperty, layoutXProperty, layoutYProperty, localToParent, localToParent, localToParent, localToParent, localToParent, localToParentTransformProperty, localToScene, localToScene, localToScene, localToScene, localToScene, localToScene, localToScene,

localToScene, localToScene, localToSceneTransformProperty, localToScreen, localToScreen, localToScreen, localToScreen, lookupAll, managedProperty, maxHeight, maxWidth, mouseTransparentProperty, nodeOrientationProperty, notifyAccessibleAttributeChanged, onContextMenuRequestedProperty, onDragDetectedProperty, onDragDoneProperty, onDragDroppedProperty, onDragEnteredProperty, onDragExitedProperty, onDragOverProperty, onInputMethodTextChangedProperty, onKeyPressedProperty, onKeyReleasedProperty, onKeyTypedProperty, onMouseClickedProperty, onMouseDragEnteredProperty, onMouseDragExitedProperty, onMouseDraggedProperty, onMouseDragOverProperty, onMouseDragReleasedProperty, onMouseEnteredProperty, onMouseExitedProperty, onMouseMovedProperty, onMousePressedProperty, onMouseReleasedProperty, onRotateProperty, onRotationFinishedProperty, onRotationStartedProperty, onScrollFinishedProperty, onScrollProperty, onScrollStartedProperty, onSwipeDownProperty, onSwipeLeftProperty, onSwipeRightProperty, onSwipeUpProperty, onTouchMovedProperty, onTouchPressedProperty, onTouchReleasedProperty, onTouchStationaryProperty, onZoomFinishedProperty, onZoomProperty, onZoomStartedProperty, opacityProperty, parentProperty, parentToLocal, parentToLocal, parentToLocal, parentToLocal, parentToLocal, pickOnBoundsProperty, pressedProperty, pseudoClassStateChanged, relocate, removeEventFilter, removeEventHandler, requestFocus, resize, resizeRelocate, rotateProperty, rotationAxisProperty, scaleXProperty, scaleYProperty, scaleZProperty, sceneProperty, sceneToLocal, sceneToLocal, sceneToLocal, sceneToLocal, sceneToLocal, sceneToLocal, sceneToLocal, sceneToLocal, screenToLocal, screenToLocal, screenToLocal, setAccessibleHelp, setAccessibleRole, setAccessibleRoleDescription, setAccessibleText, setBlendMode, setCache, setCacheHint, setClip, setCursor, setDepthTest, setDisable, setDisabled, setEffect, setEventDispatcher, setEventHandler, setFocused, setFocusTraversable, setHover, setId, setInputMethodRequests, setLayoutX, setLayoutY, setManaged, setMouseTransparent, setNodeOrientation, setOnContextMenuRequested, setOnDragDetected, setOnDragDone, setOnDragDropped, setOnDragEntered, setOnDragExited, setOnDragOver, setOnInputMethodTextChanged, setOnKeyPressed, setOnKeyReleased, setOnKeyTyped, setOnMouseClicked, setOnMouseDragEntered, setOnMouseDragExited, setOnMouseDragged, setOnMouseDragOver, setOnMouseDragReleased, setOnMouseEntered, setOnMouseExited, setOnMouseMoved, setOnMousePressed, setOnMouseReleased, setOnRotate, setOnRotationFinished, setOnRotationStarted, setOnScroll, setOnScrollFinished, setOnScrollStarted, setOnSwipeDown, setOnSwipeLeft, setOnSwipeRight, setOnSwipeUp, setOnTouchMoved, setOnTouchPressed, setOnTouchReleased, setOnTouchStationary, setOnZoom, setOnZoomFinished, setOnZoomStarted, setOpacity, setPickOnBounds, setPressed, setRotate, setRotationAxis, setScaleX, setScaleY, setScaleZ, setStyle, setTranslateX, setTranslateY, setTranslateZ, setUserData, setVisible, snapshot, snapshot, startDragAndDrop, startFullDrag, styleProperty, toBack, toFront, toString, translateXProperty, translateYProperty, translateZProperty, usesMirroring, visibleProperty

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Property Detail

autoSizeChildren

```
public final BooleanProperty autoSizeChildrenProperty
```

Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass. If set to false, then the application is responsible for setting the size of this Group's resizable children, otherwise such nodes may end up with a zero width/height and will not be visible. This variable has no effect on content nodes which are not resizable (Shape, Text, etc).

Default value:

true

See Also:

isAutoSizeChildren(), setAutoSizeChildren(boolean)

Constructor Detail

Group

```
public Group()
```

Constructs a group.

Group

```
public Group(Node... children)
```

Constructs a group consisting of children.

Parameters:

children - children.

Group

```
public Group(Collection<Node> children)
```

Constructs a group consisting of the given children.

Parameters:

children - children of the group

Throws:

NullPointerException - if the specified collection is null

Since:

JavaFX 8.0

Method Detail

setAutoSizeChildren

```
public final void setAutoSizeChildren(boolean value)
```

Sets the value of the property autoSizeChildren.

Property description:

Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass. If set to false, then the application is responsible for setting the size of this Group's resizable children, otherwise such nodes may end up with a zero width/height and will not be visible. This variable has no effect on content nodes which are not resizable (Shape, Text, etc).

Default value:

true

isAutoSizeChildren

```
public final boolean isAutoSizeChildren()
```

Gets the value of the property autoSizeChildren.

Property description:

Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass. If set to false, then the application is responsible for setting the size of this Group's resizable children, otherwise such nodes may end up with a zero width/height and will not be visible. This variable has no effect on content nodes which are not resizable (Shape, Text, etc).

Default value:

true

autoSizeChildrenProperty

```
public final BooleanProperty autoSizeChildrenProperty()
```

Controls whether or not this Group will automatically resize any managed resizable children to their preferred sizes during the layout pass. If set to false, then the

application is responsible for setting the size of this Group's resizable children, otherwise such nodes may end up with a zero width/height and will not be visible. This variable has no effect on content nodes which are not resizable (Shape, Text, etc).

Default value:

`true`

See Also:

`isAutoSizeChildren()`, `setAutoSizeChildren(boolean)`

getChildren

```
public ObservableList<Node> getChildren()
```

Gets the list of children of this `Group`.

Overrides:

`getChildren` in class `Parent`

Returns:

the list of children of this Group.

prefWidth

```
public double prefWidth(double height)
```

Group defines the preferred width as simply being the width of its layout bounds, which in turn is simply the sum of the positions & widths of all of its children. That is, the preferred width is the one that it is at, because a Group cannot be resized. Note: as the layout bounds in autosize Group depend on the Group to be already laid-out, this call will do the layout of the Group if necessary.

Overrides:

`prefWidth` in class `Parent`

Parameters:

height - This parameter is ignored by Group

Returns:

The layout bounds width

See Also:

`Node.isResizable()`, `Node.getContentBias()`, `Node.autosize()`

prefHeight

```
public double prefHeight(double width)
```

Group defines the preferred height as simply being the height of its layout bounds, which in turn is simply the sum of the positions & heights of all of its children. That is, the preferred height is the one that it is at, because a Group cannot be resized. Note: as the layout bounds in autosize Group depend on the Group to be already laid

out, this call will do the layout of the Group if necessary.

Overrides:

`prefHeight` in class `Parent`

Parameters:

`width - This parameter is ignored by Group`

Returns:

`The layout bounds height`

See Also:

`Node.getContentBias()`, `Node.autosize()`

minHeight

```
public double minHeight(double width)
```

Description copied from class: Node

Returns the node's minimum height for use in layout calculations. If the node is resizable, its parent should not resize its height any smaller than this value. If the node is not resizable, returns its layoutBounds height.

Layout code which calls this method should first check the content-bias of the node. If the node has a horizontal content-bias, then callers should pass in a width value that the minimum height should be based on. If the node has either a vertical or null content-bias, then the caller should pass in -1.

Node subclasses with a horizontal content-bias should honor the width parameter whether -1 or a positive value. All other subclasses may ignore the width parameter (which will likely be -1).

If Node's `Node.maxHeight(double)` is lower than this number, `minHeight` takes precedence. This means the Node should never be resized below `minHeight`.

Overrides:

`minHeight` in class `Parent`

Parameters:

`width - the width that should be used if minimum height depends on it`

Returns:

`the minimum height that the node should be resized to during layout The result will never be NaN, nor will it ever be negative.`

See Also:

`Node.isResizable()`, `Node.getContentBias()`

minWidth

```
public double minWidth(double height)
```

Description copied from class: Node

Returns the node's minimum width for use in layout calculations. If the node is

resizable, its parent should not resize its width any smaller than this value. If the node is not resizable, returns its layoutBounds width.

Layout code which calls this method should first check the content-bias of the node. If the node has a vertical content-bias, then callers should pass in a height value that the minimum width should be based on. If the node has either a horizontal or null content-bias, then the caller should pass in -1.

Node subclasses with a vertical content-bias should honor the height parameter whether -1 or a positive value. All other subclasses may ignore the height parameter (which will likely be -1).

If Node's `Node.maxWidth(double)` is lower than this number, `minWidth` takes precedence. This means the Node should never be resized below `minWidth`.

Overrides:

`minWidth` in class `Parent`

Parameters:

`height - the height that should be used if minimum width depends on it`

Returns:

`the minimum width that the node should be resized to during layout. The result will never be NaN, nor will it ever be negative.`

See Also:

`Node.isResizable()`, `Node.getContentBias()`

layoutChildren

```
protected void layoutChildren()
```

Group implements layoutChildren such that each child is resized to its preferred size, if the child is resizable. Non-resizable children are simply left alone. If `autoSizeChildren` is false, then Group does nothing in this method.

Overrides:

`layoutChildren` in class `Parent`