public class

Summary: Nested Classes | XML Attrs | Inherited XML Attrs | Constants | Inherited Constants | Inherited Fields | Ctors | Methods | Protected Methods | Inherited Methods | [Expand All]

Added in API level 1

LinearLayout

extends ViewGroup

java.lang.Object
↳ android.view.View
↳ android.view.ViewGroup
↳ android.widget.LinearLayout

Known Direct Subclasses
NumberPicker, RadioGroup, SearchView, TabWidget, TableLayout, TableRow, ZoomControls

Class Overview

A Layout that arranges its children in a single column or a single row. The direction of the row can be set by calling `setOrientation()` (/reference/android/widget/LinearLayout.html#setOrientation(int)). You can also specify gravity, which specifies the alignment of all the child elements by calling `setGravity()` (/reference/android/widget/LinearLayout.html#setGravity(int)) or specify that specific children grow to fill up any remaining space in the layout by setting the *weight* member of `LinearLayout.LayoutParams` (/reference/android/widget/LinearLayout.LayoutParams.html). The default orientation is horizontal.

See the Linear Layout (/guide/topics/ui/layout/linear.html) guide.

Also see `android.widget.LinearLayout.LayoutParams` (/reference/android/widget/LinearLayout.LayoutParams.html) for layout attributes

Summary

Nested Classes

class LinearLayout.LayoutParams	Per-child layout information associated with ViewLinearLayout.

XML Attributes

Attribute Name	*Related Method*	*Description*
android:baselineAligned	setBaselineAligned(boolean)	When set to false, prevents

		the layout from aligning its children's baselines.
android:baselineAlignedChildIndex	setBaselineAlignedChildIndex(int)	When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).
android:divider	setDividerDrawable(Drawable)	Drawable to use as a vertical divider between buttons.
android:gravity	setGravity(int)	Specifies how an object should position its content, on both the X and Y axes, within its own bounds.
android:measureWithLargestChild	setMeasureWithLargestChildEnabled(boolean)	When set to true, all children with a weight will be considered having the minimum size of the largest

		child.
android:orientation	setOrientation(int)	Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column.
android:weightSum		Defines the maximum weight sum.

Inherited XML Attributes [Expand]

- From class android.view.ViewGroup
- From class android.view.View

Constants

int HORIZONTAL	
int SHOW_DIVIDER_BEGINNING	Show a divider at the beginning of the group.
int SHOW_DIVIDER_END	Show a divider at the end of the group.
int SHOW_DIVIDER_MIDDLE	Show dividers between each item in the group.
int SHOW_DIVIDER_NONE	Don't show any dividers.
int VERTICAL	

Inherited Constants [Expand]

- From class android.view.ViewGroup
- From class android.view.View

Inherited Fields [Expand]

- From class android.view.View

Public Constructors

LinearLayout (Context context)
LinearLayout (Context context, AttributeSet attrs)
LinearLayout (Context context, AttributeSet attrs, int defStyle)

Public Methods

LinearLayout.LayoutParams	generateLayoutParams (AttributeSet attrs) Returns a new set of layout parameters based on the supplied attributes set.
int	getBaseline () Return the offset of the widget's text baseline from the widget's top boundary.

int	getBaselineAlignedChildIndex ()
Drawable	getDividerDrawable ()
int	getDividerPadding () Get the padding size used to inset dividers in pixels
int	getOrientation () Returns the current orientation.
int	getShowDividers ()
float	getWeightSum () Returns the desired weights sum.
boolean	isBaselineAligned () Indicates whether widgets contained within this layout are aligned on their baseline or not.
boolean	isMeasureWithLargestChildEnabled () When true, all children with a weight will be considered having the minimum size of the largest child.
void	onInitializeAccessibilityEvent (AccessibilityEvent event) Initializes an `AccessibilityEvent` with information about this View which is the event source.
void	onInitializeAccessibilityNodeInfo (AccessibilityNodeInfo info) Initializes an `AccessibilityNodeInfo` with information about this view.
void	setBaselineAligned (boolean baselineAligned) Defines whether widgets contained in this layout are baseline-aligned or not.
void	setBaselineAlignedChildIndex (int i)
void	setDividerDrawable (Drawable divider) Set a drawable to be used as a divider between items.
void	setDividerPadding (int padding) Set padding displayed on both ends of dividers.
void	setGravity (int gravity) Describes how the child views are positioned.
void	setHorizontalGravity (int horizontalGravity)
void	setMeasureWithLargestChildEnabled (boolean enabled) When set to true, all children with a weight will be considered having the minimum size of the largest child.
void	setOrientation (int orientation) Should the layout be a column or a row.
void	setShowDividers (int showDividers) Set how dividers should be shown between items in this layout

void	setVerticalGravity (int verticalGravity)
void	setWeightSum (float weightSum) Defines the desired weights sum.
boolean	shouldDelayChildPressedState () Return true if the pressed state should be delayed for children or descendants of this ViewGroup.

Protected Methods

boolean	checkLayoutParams (ViewGroup.LayoutParams p)
LinearLayout.LayoutParams	generateDefaultLayoutParams () Returns a set of layout parameters with a width of `MATCH_PARENT` and a height of `WRAP_CONTENT` when the layout's orientation is `VERTICAL`.
LinearLayout.LayoutParams	generateLayoutParams (ViewGroup.LayoutParams p) Returns a safe set of layout parameters based on the supplied layout params.
void	onDraw (Canvas canvas) Implement this to do your drawing.
void	onLayout (boolean changed, int l, int t, int r, int b) Called from layout when this view should assign a size and position to each of its children.
void	onMeasure (int widthMeasureSpec, int heightMeasureSpec) Measure the view and its content to determine the measured width and the measured height.

Inherited Methods [Expand]

- From class android.view.ViewGroup
- From class android.view.View
- From class java.lang.Object
- From interface android.graphics.drawable.Drawable.Callback
- From interface android.view.KeyEvent.Callback
- From interface android.view.ViewManager
- From interface android.view.ViewParent
- From interface android.view.accessibility.AccessibilityEventSource

XML Attributes

android:baselineAligned

When set to false, prevents the layout from aligning its children's baselines. This attribute is particularly useful when the children use different values for gravity. The default value is true.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:][`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `baselineAligned` (/reference/android/R.attr.html#baselineAligned).

Related Methods

setBaselineAligned(boolean)

android:baselineAlignedChildIndex

When a linear layout is part of another layout that is baseline aligned, it can specify which of its children to baseline align to (that is, which child TextView).

Must be an integer value, such as "`100`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:][`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `baselineAlignedChildIndex` (/reference/android/R.attr.html#baselineAlignedChildIndex).

Related Methods

setBaselineAlignedChildIndex(int)

android:divider

Drawable to use as a vertical divider between buttons.

May be a reference to another resource, in the form "`@[+][`*package*`:]`*type*`:`*name*" or to a theme attribute in the form "`?[`*package*`:][`*type*`:]`*name*".

May be a color value, in the form of "*`#rgb`*", "*`#argb`*", "*`#rrggbb`*", or "*`#aarrggbb`*".

This corresponds to the global attribute resource symbol `divider` (/reference/android/R.attr.html#divider).

Related Methods

setDividerDrawable(Drawable)

android:gravity

Specifies how an object should position its content, on both the X and Y axes, within its own bounds.

Must be one or more (separated by '|') of the following constant values.

Constant	Value	Description
top	0x30	Push object to the top of its container, not changing its size.
bottom	0x50	Push object to the bottom of its container, not changing its size.
left	0x03	Push object to the left of its container, not changing its size.
right	0x05	Push object to the right of its container, not changing its size.
center_vertical	0x10	Place object in the vertical center of its container, not changing its size.
fill_vertical	0x70	Grow the vertical size of the object if needed so it completely fills its container.
center_horizontal	0x01	Place object in the horizontal center of its container, not changing its size.
fill_horizontal	0x07	Grow the horizontal size of the object if needed so it completely fills its container.
center	0x11	Place the object in the center of its container in both the vertical and horizontal axis, not changing its size.
fill	0x77	Grow the horizontal and vertical size of the object if needed so it completely fills its container.
clip_vertical	0x80	Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges.
clip_horizontal	0x08	Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will

		be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges.
`start`	0x00800003	Push object to the beginning of its container, not changing its size.
`end`	0x00800005	Push object to the end of its container, not changing its size.

This corresponds to the global attribute resource symbol `gravity` (/reference/android/R.attr.html#gravity).

Related Methods

setGravity(int)

android:measureWithLargestChild

When set to true, all children with a weight will be considered having the minimum size of the largest child. If false, all children are measured normally.

Must be a boolean value, either "`true`" or "`false`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:][`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `measureWithLargestChild` (/reference/android/R.attr.html#measureWithLargestChild).

Related Methods

setMeasureWithLargestChildEnabled(boolean)

android:orientation

Should the layout be a column or a row? Use "horizontal" for a row, "vertical" for a column. The default is horizontal.

Must be one of the following constant values.

Constant	Value	Description
`horizontal`	0	Defines an horizontal widget.
`vertical`	1	Defines a vertical widget.

This corresponds to the global attribute resource symbol `orientation` (/reference/android/R.attr.html#orientation).

Related Methods
setOrientation(int)

android:weightSum

Defines the maximum weight sum. If unspecified, the sum is computed by adding the layout_weight of all of the children. This can be used for instance to give a single child 50% of the total available space by giving it a layout_weight of 0.5 and setting the weightSum to 1.0.

Must be a floating point value, such as "`1.2`".

This may also be a reference to a resource (in the form "`@[`*package*`:]`*type*`:`*name*") or theme attribute (in the form "`?[`*package*`:][`*type*`:]`*name*") containing a value of this type.

This corresponds to the global attribute resource symbol `weightSum` (/reference/android/R.attr.html#weightSum).

Related Methods

Constants

public static final int **HORIZONTAL** Added in API level 1

Constant Value: 0 (0x00000000)

public static final int **SHOW_DIVIDER_BEGINNING** Added in API level 11

Show a divider at the beginning of the group.

Constant Value: 1 (0x00000001)

public static final int **SHOW_DIVIDER_END** Added in API level 11

Show a divider at the end of the group.

Constant Value: 4 (0x00000004)

public static final int **SHOW_DIVIDER_MIDDLE** Added in API level 11

Show dividers between each item in the group.

Constant Value: 2 (0x00000002)

public static final int **SHOW_DIVIDER_NONE** Added in API level 11

Don't show any dividers.

Constant Value: 0 (0x00000000)

public static final int **VERTICAL** Added in API level 1

Constant Value: 1 (0x00000001)

Public Constructors

public **LinearLayout** (Context context) Added in API level 1

public **LinearLayout** (Context context, AttributeSet attrs) Added in API level 1

public **LinearLayout** (Context context, AttributeSet attrs, int defStyle) Added in API level 11

Public Methods

public LinearLayout.LayoutParams **generateLayoutParams** (AttributeSet attrs) Added in API level 1

Returns a new set of layout parameters based on the supplied attributes set.

Parameters

attrs the attributes to build the layout parameters from

Returns

an instance of `ViewGroup.LayoutParams` or one of its descendants

public int **getBaseline** () Added in API level 1

Return the offset of the widget's text baseline from the widget's top boundary. If this widget does not support baseline alignment, this method returns -1.

Returns

the offset of the baseline within the widget's bounds or -1 if baseline alignment is not supported

public int **getBaselineAlignedChildIndex** () Added in API level 1

Returns

The index of the child that will be used if this layout is part of a larger layout that is baseline aligned, or -1 if none has been set.

public Drawable **getDividerDrawable** ()

Added in API level 16

Related XML Attributes

android:divider

Returns

the divider Drawable that will divide each item.

See Also

`setDividerDrawable(Drawable)`

public int **getDividerPadding** ()

Added in API level 14

Get the padding size used to inset dividers in pixels

See Also

`setShowDividers(int)`
`setDividerDrawable(Drawable)`
`setDividerPadding(int)`

public int **getOrientation** ()

Added in API level 1

Returns the current orientation.

Returns

either HORIZONTAL or VERTICAL

public int **getShowDividers** ()

Added in API level 11

Returns

A flag set indicating how dividers should be shown around items.

See Also

`setShowDividers(int)`

public float **getWeightSum** ()

Added in API level 1

Returns the desired weights sum.

Returns

A number greater than 0.0f if the weight sum is defined, or a number lower than or equals to 0.0f if not weight sum is to be used.

public boolean **isBaselineAligned** ()

Added in API level 1

Indicates whether widgets contained within this layout are aligned on

their baseline or not.

Returns

true when widgets are baseline-aligned, false otherwise

public boolean **isMeasureWithLargestChildEnabled** ()

Added in API level 11

When true, all children with a weight will be considered having the minimum size of the largest child. If false, all children are measured normally.

Related XML Attributes

android:measureWithLargestChild

Returns

True to measure children with a weight using the minimum size of the largest child, false otherwise.

public void **onInitializeAccessibilityEvent** (AccessibilityEvent event)

Added in API level 14

Initializes an `AccessibilityEvent (/reference/android/view/accessibility/AccessibilityEvent.html)` with information about this View which is the event source. In other words, the source of an accessibility event is the view whose state change triggered firing the event.

Example: Setting the password property of an event in addition to properties set by the super implementation:

```
public void onInitializeAccessibilityEvent(Accessibi
    super.onInitializeAccessibilityEvent(event);
    event.setPassword(true);
}
```

If an `View.AccessibilityDelegate (/reference/android/view/View.AccessibilityDelegate.html)` has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate) (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate))` its `onInitializeAccessibilityEvent(View, AccessibilityEvent) (/reference/android/view/View.AccessibilityDelegate.html#onInitializeAccessibilityEvent(android.view.View, android.view.accessibility.AccessibilityEvent))` is responsible for handling this call.

> **Note:** Always call the super implementation before adding information to the event, in case the default implementation has

basic information to add.

Parameters

event The event to initialize.

public void **onInitializeAccessibilityNodeInfo** (AccessibilityNodeInfo info)

Added in API level 14

Initializes an `AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo.html)` with information about this view. The base implementation sets:

- `setParent(View)`,
- `setBoundsInParent(Rect)`,
- `setBoundsInScreen(Rect)`,
- `setPackageName(CharSequence)`,
- `setClassName(CharSequence)`,
- `setContentDescription(CharSequence)`,
- `setEnabled(boolean)`,
- `setClickable(boolean)`,
- `setFocusable(boolean)`,
- `setFocused(boolean)`,
- `setLongClickable(boolean)`,
- `setSelected(boolean)`,

Subclasses should override this method, call the super implementation, and set additional attributes.

If an `View.AccessibilityDelegate (/reference/android/view/View.AccessibilityDelegate.html)` has been specified via calling `setAccessibilityDelegate(AccessibilityDelegate) (/reference/android/view/View.html#setAccessibilityDelegate(android.view.View.AccessibilityDelegate))` its `onInitializeAccessibilityNodeInfo(View, AccessibilityNodeInfo) (/reference/android/view/View.AccessibilityDelegate.html#onInitializeAccessibilityNodeInfo(android.view.View, android.view.accessibility.AccessibilityNodeInfo))` is responsible for handling this call.

Parameters

info The instance to initialize.

public void **setBaselineAligned** (boolean baselineAligned)

Added in API level 1

Defines whether widgets contained in this layout are baseline-aligned or

not.

Related XML Attributes

android:baselineAligned

Parameters

baselineAligned	true to align widgets on their baseline, false otherwise

public void **setBaselineAlignedChildIndex** (int i) Added in API level 1

Related XML Attributes

android:baselineAlignedChildIndex

Parameters

i	The index of the child that will be used if this layout is part of a larger layout that is baseline aligned.

public void **setDividerDrawable** (Drawable divider) Added in API level 11

Set a drawable to be used as a divider between items.

Related XML Attributes

android:divider

Parameters

divider	Drawable that will divide each item.

See Also

`setShowDividers(int)`

public void **setDividerPadding** (int padding) Added in API level 14

Set padding displayed on both ends of dividers.

Parameters

padding	Padding value in pixels that will be applied to each end

See Also

`setShowDividers(int)`
`setDividerDrawable(Drawable)`
`getDividerPadding()`

public void **setGravity** (int gravity) Added in API level 1

Describes how the child views are positioned. Defaults to GRAVITY_TOP. If this layout has a VERTICAL orientation, this controls where all the child views are placed if there is extra vertical space. If this layout has a HORIZONTAL orientation, this controls the alignment

of the children.

Related XML Attributes

android:gravity

Parameters

gravity See `Gravity`

public void **setHorizontalGravity** (int horizontalGravity) Added in API level 1

public void **setMeasureWithLargestChildEnabled** (boolean enabled) Added in API level 11

When set to true, all children with a weight will be considered having the minimum size of the largest child. If false, all children are measured normally. Disabled by default.

Related XML Attributes

android:measureWithLargestChild

Parameters

enabled True to measure children with a weight using the minimum size of the largest child, false otherwise.

public void **setOrientation** (int orientation) Added in API level 1

Should the layout be a column or a row.

Related XML Attributes

android:orientation

Parameters

orientation Pass HORIZONTAL or VERTICAL. Default value is HORIZONTAL.

public void **setShowDividers** (int showDividers) Added in API level 11

Set how dividers should be shown between items in this layout

Parameters

showDividers One or more of `SHOW_DIVIDER_BEGINNING`, `SHOW_DIVIDER_MIDDLE`, or `SHOW_DIVIDER_END`, or `SHOW_DIVIDER_NONE` to show no dividers.

public void **setVerticalGravity** (int verticalGravity) Added in API level 1

public void **setWeightSum** (float weightSum) Added in API level 1

Defines the desired weights sum. If unspecified the weights sum is computed at layout time by adding the layout_weight of each child. This can be used for instance to give a single child 50% of the total available space by giving it a layout_weight of 0.5 and setting the weightSum to 1.0.

Parameters

weightSum	a number greater than 0.0f, or a number lower than or equals to 0.0f if the weight sum should be computed from the children's layout_weight

public boolean **shouldDelayChildPressedState** () Added in API level 14

Return true if the pressed state should be delayed for children or descendants of this ViewGroup. Generally, this should be done for containers that can scroll, such as a List. This prevents the pressed state from appearing when the user is actually trying to scroll the content. The default implementation returns true for compatibility reasons. Subclasses that do not scroll should generally override this method and return false.

Protected Methods

protected boolean **checkLayoutParams** (ViewGroup.LayoutParams p) Added in API level 1

protected LinearLayout.LayoutParams **generateDefaultLayoutParams** () Added in API level 1

Returns a set of layout parameters with a width of MATCH_PARENT (/reference/android/view/ViewGroup.LayoutParams.html#MATCH_PARENT) and a height of WRAP_CONTENT (/reference/android/view/ViewGroup.LayoutParams.html#WRAP_CONTENT) when the layout's orientation is VERTICAL (/reference/android/widget/LinearLayout.html#VERTICAL). When the orientation is HORIZONTAL (/reference/android/widget/LinearLayout.html#HORIZONTAL), the width is set to WRAP_CONTENT (/reference/android/view/ViewGroup.LayoutParams.html#WRAP_CONTENT) and the height to WRAP_CONTENT (/reference/android/view/ViewGroup.LayoutParams.html#WRAP_CONTENT).

Returns

a set of default layout parameters or null

protected LinearLayout.LayoutParams
generateLayoutParams (ViewGroup.LayoutParams p) Added in API level 1

Returns a safe set of layout parameters based on the supplied layout params. When a ViewGroup is passed a View whose layout params do not pass the test of `checkLayoutParams(android.view.ViewGroup.LayoutParams)` (/reference/android/view/ViewGroup.html#checkLayoutParams(android.view.ViewGroup.LayoutParams)), this method is invoked. This method should return a new set of layout params suitable for this ViewGroup, possibly by copying the appropriate attributes from the specified set of layout params.

Parameters

p The layout parameters to convert into a suitable set of layout parameters for this ViewGroup.

Returns

an instance of `ViewGroup.LayoutParams` or one of its descendants

protected void **onDraw** (Canvas canvas) Added in API level 1

Implement this to do your drawing.

Parameters

canvas the canvas on which the background will be drawn

protected void **onLayout** (boolean changed, int l, int t, int r, int b) Added in API level 1

Called from layout when this view should assign a size and position to each of its children. Derived classes with children should override this method and call layout on each of their children.

Parameters

changed This is a new size or position for this view

l Left position, relative to parent

t Top position, relative to parent

r Right position, relative to parent

b Bottom position, relative to parent

protected void **onMeasure** (int widthMeasureSpec, int heightMeasureSpec) Added in API level 1

Measure the view and its content to determine the measured width and the measured height. This method is invoked by `measure(int, int)`

(/reference/android/view/View.html#measure(int, int)) and should be overriden by subclasses to provide accurate and efficient measurement of their contents.

CONTRACT: When overriding this method, you *must* call setMeasuredDimension(int, int) (/reference/android/view/View.html#setMeasuredDimension(int, int)) to store the measured width and height of this view. Failure to do so will trigger an IllegalStateException, thrown by measure(int, int) (/reference/android/view/View.html#measure(int, int)). Calling the superclass' onMeasure(int, int) (/reference/android/view/View.html#onMeasure(int, int)) is a valid use.

The base class implementation of measure defaults to the background size, unless a larger size is allowed by the MeasureSpec. Subclasses should override onMeasure(int, int) (/reference/android/view/View.html#onMeasure(int, int)) to provide better measurements of their content.

If this method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (getSuggestedMinimumHeight() (/reference/android/view/View.html#getSuggestedMinimumHeight()) and getSuggestedMinimumWidth() (/reference/android/view/View.html#getSuggestedMinimumWidth())).

Parameters

widthMeasureSpec	horizontal space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec.
heightMeasureSpec	vertical space requirements as imposed by the parent. The requirements are encoded with View.MeasureSpec.