| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicScrollBarUI.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicRootPaneUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicScrollBarUI.html)    [**NO FRAMES**](http://docs.google.com/BasicScrollBarUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#32hioqz) | [METHOD](#41mghml) |

## **javax.swing.plaf.basic**

Class BasicScrollBarUI

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.plaf.ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html)  
 [javax.swing.plaf.ScrollBarUI](http://docs.google.com/javax/swing/plaf/ScrollBarUI.html)  
 **javax.swing.plaf.basic.BasicScrollBarUI**

**All Implemented Interfaces:** [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html) **Direct Known Subclasses:** [MetalScrollBarUI](http://docs.google.com/javax/swing/plaf/metal/MetalScrollBarUI.html)

public class **BasicScrollBarUI**extends [ScrollBarUI](http://docs.google.com/javax/swing/plaf/ScrollBarUI.html)implements [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html), [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html)

Implementation of ScrollBarUI for the Basic Look and Feel

| **Nested Class Summary** | |
| --- | --- |
| protected  class | [**BasicScrollBarUI.ArrowButtonListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html)            Listener for cursor keys. |
| protected  class | [**BasicScrollBarUI.ModelListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ModelListener.html)            A listener to listen for model changes. |
| class | [**BasicScrollBarUI.PropertyChangeHandler**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.PropertyChangeHandler.html) |
| protected  class | [**BasicScrollBarUI.ScrollListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ScrollListener.html)            Listener for scrolling events initiated in the ScrollPane. |
| protected  class | [**BasicScrollBarUI.TrackListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.TrackListener.html)            Track mouse drags. |

| **Field Summary** | |
| --- | --- |
| protected  [BasicScrollBarUI.ArrowButtonListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) | [**buttonListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#buttonListener) |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**decrButton**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#decrButton) |
| protected static int | [**DECREASE\_HIGHLIGHT**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#DECREASE_HIGHLIGHT) |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**incrButton**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#incrButton) |
| protected static int | [**INCREASE\_HIGHLIGHT**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#INCREASE_HIGHLIGHT) |
| protected  boolean | [**isDragging**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#isDragging) |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**maximumThumbSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#maximumThumbSize) |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**minimumThumbSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#minimumThumbSize) |
| protected  [BasicScrollBarUI.ModelListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ModelListener.html) | [**modelListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#modelListener) |
| protected static int | [**NO\_HIGHLIGHT**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#NO_HIGHLIGHT) |
| protected  [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) | [**propertyChangeListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#propertyChangeListener) |
| protected  [JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) | [**scrollbar**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#scrollbar) |
| protected  [BasicScrollBarUI.ScrollListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ScrollListener.html) | [**scrollListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#scrollListener) |
| protected  [Timer](http://docs.google.com/javax/swing/Timer.html) | [**scrollTimer**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#scrollTimer) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**thumbColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#thumbColor) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**thumbDarkShadowColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#thumbDarkShadowColor) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**thumbHighlightColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#thumbHighlightColor) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**thumbLightShadowColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#thumbLightShadowColor) |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**thumbRect**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#thumbRect) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**trackColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#trackColor) |
| protected  int | [**trackHighlight**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#trackHighlight) |
| protected  [Color](http://docs.google.com/java/awt/Color.html) | [**trackHighlightColor**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#trackHighlightColor) |
| protected  [BasicScrollBarUI.TrackListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.TrackListener.html) | [**trackListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#trackListener) |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**trackRect**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#trackRect) |

| **Fields inherited from interface javax.swing.**[**SwingConstants**](http://docs.google.com/javax/swing/SwingConstants.html) |
| --- |
| [BOTTOM](http://docs.google.com/javax/swing/SwingConstants.html#BOTTOM), [CENTER](http://docs.google.com/javax/swing/SwingConstants.html#CENTER), [EAST](http://docs.google.com/javax/swing/SwingConstants.html#EAST), [HORIZONTAL](http://docs.google.com/javax/swing/SwingConstants.html#HORIZONTAL), [LEADING](http://docs.google.com/javax/swing/SwingConstants.html#LEADING), [LEFT](http://docs.google.com/javax/swing/SwingConstants.html#LEFT), [NEXT](http://docs.google.com/javax/swing/SwingConstants.html#NEXT), [NORTH](http://docs.google.com/javax/swing/SwingConstants.html#NORTH), [NORTH\_EAST](http://docs.google.com/javax/swing/SwingConstants.html#NORTH_EAST), [NORTH\_WEST](http://docs.google.com/javax/swing/SwingConstants.html#NORTH_WEST), [PREVIOUS](http://docs.google.com/javax/swing/SwingConstants.html#PREVIOUS), [RIGHT](http://docs.google.com/javax/swing/SwingConstants.html#RIGHT), [SOUTH](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH), [SOUTH\_EAST](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH_EAST), [SOUTH\_WEST](http://docs.google.com/javax/swing/SwingConstants.html#SOUTH_WEST), [TOP](http://docs.google.com/javax/swing/SwingConstants.html#TOP), [TRAILING](http://docs.google.com/javax/swing/SwingConstants.html#TRAILING), [VERTICAL](http://docs.google.com/javax/swing/SwingConstants.html#VERTICAL), [WEST](http://docs.google.com/javax/swing/SwingConstants.html#WEST) |

| **Constructor Summary** | |
| --- | --- |
| [**BasicScrollBarUI**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#BasicScrollBarUI())() |

| **Method Summary** | |
| --- | --- |
| void | [**addLayoutComponent**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#addLayoutComponent(java.lang.String,%20java.awt.Component))([String](http://docs.google.com/java/lang/String.html) name, [Component](http://docs.google.com/java/awt/Component.html) child)            If the layout manager uses a per-component string, adds the component comp to the layout, associating it with the string specified by name. |
| protected  void | [**configureScrollBarColors**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#configureScrollBarColors())() |
| protected  [BasicScrollBarUI.ArrowButtonListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) | [**createArrowButtonListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createArrowButtonListener())() |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**createDecreaseButton**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createDecreaseButton(int))(int orientation) |
| protected  [JButton](http://docs.google.com/javax/swing/JButton.html) | [**createIncreaseButton**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createIncreaseButton(int))(int orientation) |
| protected  [BasicScrollBarUI.ModelListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ModelListener.html) | [**createModelListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createModelListener())() |
| protected  [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) | [**createPropertyChangeListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createPropertyChangeListener())() |
| protected  [BasicScrollBarUI.ScrollListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ScrollListener.html) | [**createScrollListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createScrollListener())() |
| protected  [BasicScrollBarUI.TrackListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.TrackListener.html) | [**createTrackListener**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createTrackListener())() |
| static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) | [**createUI**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#createUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c) |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMaximumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Returns the specified component's maximum size appropriate for the look and feel. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumThumbSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMaximumThumbSize())()            Return the largest acceptable size for the thumb. |
| protected  [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMinimumThumbSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMinimumThumbSize())()            Return the smallest acceptable size for the thumb. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getPreferredSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            A vertical scrollbar's preferred width is the maximum of preferred widths of the (non null) increment/decrement buttons, and the minimum width of the thumb. |
| boolean | [**getSupportsAbsolutePositioning**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getSupportsAbsolutePositioning())()            Indicates whether the user can absolutely position the thumb with a mouse gesture (usually the middle mouse button). |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getThumbBounds**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getThumbBounds())()            Return the current size/location of the thumb. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getTrackBounds**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getTrackBounds())()            Returns the current bounds of the track, i.e. |
| protected  void | [**installComponents**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#installComponents())() |
| protected  void | [**installDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#installDefaults())() |
| protected  void | [**installKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#installKeyboardActions())() |
| protected  void | [**installListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#installListeners())() |
| void | [**installUI**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#installUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Configures the specified component appropriate for the look and feel. |
| boolean | [**isThumbRollover**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#isThumbRollover())()            Returns true if the mouse is currently over the thumb. |
| void | [**layoutContainer**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#layoutContainer(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)            Lays out the specified container. |
| protected  void | [**layoutHScrollbar**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#layoutHScrollbar(javax.swing.JScrollBar))([JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) sb) |
| protected  void | [**layoutVScrollbar**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#layoutVScrollbar(javax.swing.JScrollBar))([JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) sb) |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**minimumLayoutSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#minimumLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)            Calculates the minimum size dimensions for the specified container, given the components it contains. |
| void | [**paint**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Paints the specified component appropriate for the look and feel. |
| protected  void | [**paintDecreaseHighlight**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#paintDecreaseHighlight(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g) |
| protected  void | [**paintIncreaseHighlight**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#paintIncreaseHighlight(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g) |
| protected  void | [**paintThumb**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#paintThumb(java.awt.Graphics,%20javax.swing.JComponent,%20java.awt.Rectangle))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) thumbBounds) |
| protected  void | [**paintTrack**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#paintTrack(java.awt.Graphics,%20javax.swing.JComponent,%20java.awt.Rectangle))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [JComponent](http://docs.google.com/javax/swing/JComponent.html) c, [Rectangle](http://docs.google.com/java/awt/Rectangle.html) trackBounds) |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**preferredLayoutSize**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#preferredLayoutSize(java.awt.Container))([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)            Calculates the preferred size dimensions for the specified container, given the components it contains. |
| void | [**removeLayoutComponent**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#removeLayoutComponent(java.awt.Component))([Component](http://docs.google.com/java/awt/Component.html) child)            Removes the specified component from the layout. |
| protected  void | [**scrollByBlock**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#scrollByBlock(int))(int direction) |
| protected  void | [**scrollByUnit**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#scrollByUnit(int))(int direction) |
| protected  void | [**setThumbBounds**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#setThumbBounds(int,%20int,%20int,%20int))(int x, int y, int width, int height)            Set the bounds of the thumb and force a repaint that includes the old thumbBounds and the new one. |
| protected  void | [**setThumbRollover**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#setThumbRollover(boolean))(boolean active)            Sets whether or not the mouse is currently over the thumb. |
| protected  void | [**uninstallComponents**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#uninstallComponents())() |
| protected  void | [**uninstallDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#uninstallDefaults())() |
| protected  void | [**uninstallKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#uninstallKeyboardActions())() |
| protected  void | [**uninstallListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#uninstallListeners())() |
| void | [**uninstallUI**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#uninstallUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Reverses configuration which was done on the specified component during installUI. |

| **Methods inherited from class javax.swing.plaf.**[**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html) |
| --- |
| [contains](http://docs.google.com/javax/swing/plaf/ComponentUI.html#contains(javax.swing.JComponent,%20int,%20int)), [getAccessibleChild](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChild(javax.swing.JComponent,%20int)), [getAccessibleChildrenCount](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getAccessibleChildrenCount(javax.swing.JComponent)), [getBaseline](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaseline(javax.swing.JComponent,%20int,%20int)), [getBaselineResizeBehavior](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getBaselineResizeBehavior(javax.swing.JComponent)), [getMinimumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent)), [update](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### minimumThumbSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **minimumThumbSize**

### maximumThumbSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **maximumThumbSize**

### thumbHighlightColor

protected [Color](http://docs.google.com/java/awt/Color.html) **thumbHighlightColor**

### thumbLightShadowColor

protected [Color](http://docs.google.com/java/awt/Color.html) **thumbLightShadowColor**

### thumbDarkShadowColor

protected [Color](http://docs.google.com/java/awt/Color.html) **thumbDarkShadowColor**

### thumbColor

protected [Color](http://docs.google.com/java/awt/Color.html) **thumbColor**

### trackColor

protected [Color](http://docs.google.com/java/awt/Color.html) **trackColor**

### trackHighlightColor

protected [Color](http://docs.google.com/java/awt/Color.html) **trackHighlightColor**

### scrollbar

protected [JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) **scrollbar**

### incrButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **incrButton**

### decrButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **decrButton**

### isDragging

protected boolean **isDragging**

### trackListener

protected [BasicScrollBarUI.TrackListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.TrackListener.html) **trackListener**

### buttonListener

protected [BasicScrollBarUI.ArrowButtonListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) **buttonListener**

### modelListener

protected [BasicScrollBarUI.ModelListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ModelListener.html) **modelListener**

### thumbRect

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **thumbRect**

### trackRect

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **trackRect**

### trackHighlight

protected int **trackHighlight**

### NO\_HIGHLIGHT

protected static final int **NO\_HIGHLIGHT**

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.plaf.basic.BasicScrollBarUI.NO_HIGHLIGHT)

### DECREASE\_HIGHLIGHT

protected static final int **DECREASE\_HIGHLIGHT**

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.plaf.basic.BasicScrollBarUI.DECREASE_HIGHLIGHT)

### INCREASE\_HIGHLIGHT

protected static final int **INCREASE\_HIGHLIGHT**

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#javax.swing.plaf.basic.BasicScrollBarUI.INCREASE_HIGHLIGHT)

### scrollListener

protected [BasicScrollBarUI.ScrollListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ScrollListener.html) **scrollListener**

### propertyChangeListener

protected [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) **propertyChangeListener**

### scrollTimer

protected [Timer](http://docs.google.com/javax/swing/Timer.html) **scrollTimer**

| **Constructor Detail** |
| --- |

### BasicScrollBarUI

public **BasicScrollBarUI**()

| **Method Detail** |
| --- |

### createUI

public static [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **createUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

### configureScrollBarColors

protected void **configureScrollBarColors**()

### installUI

public void **installUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) Configures the specified component appropriate for the look and feel. This method is invoked when the ComponentUI instance is being installed as the UI delegate on the specified component. This method should completely configure the component for the look and feel, including the following:

1. Install any default property values for color, fonts, borders, icons, opacity, etc. on the component. Whenever possible, property values initialized by the client program should *not* be overridden.
2. Install a LayoutManager on the component if necessary.
3. Create/add any required sub-components to the component.
4. Create/install event listeners on the component.
5. Create/install a PropertyChangeListener on the component in order to detect and respond to component property changes appropriately.
6. Install keyboard UI (mnemonics, traversal, etc.) on the component.
7. Initialize any appropriate instance data.

**Overrides:**[installUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component where this UI delegate is being installed**See Also:**[ComponentUI.uninstallUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)), [JComponent.setUI(javax.swing.plaf.ComponentUI)](http://docs.google.com/javax/swing/JComponent.html#setUI(javax.swing.plaf.ComponentUI)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### uninstallUI

public void **uninstallUI**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) Reverses configuration which was done on the specified component during installUI. This method is invoked when this UIComponent instance is being removed as the UI delegate for the specified component. This method should undo the configuration performed in installUI, being careful to leave the JComponent instance in a clean state (no extraneous listeners, look-and-feel-specific property objects, etc.). This should include the following:

1. Remove any UI-set borders from the component.
2. Remove any UI-set layout managers on the component.
3. Remove any UI-added sub-components from the component.
4. Remove any UI-added event/property listeners from the component.
5. Remove any UI-installed keyboard UI from the component.
6. Nullify any allocated instance data objects to allow for GC.

**Overrides:**[uninstallUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the component from which this UI delegate is being removed; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.installUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent)), [JComponent.updateUI()](http://docs.google.com/javax/swing/JComponent.html#updateUI())

### installDefaults

protected void **installDefaults**()

### installComponents

protected void **installComponents**()

### uninstallComponents

protected void **uninstallComponents**()

### installListeners

protected void **installListeners**()

### installKeyboardActions

protected void **installKeyboardActions**()

### uninstallKeyboardActions

protected void **uninstallKeyboardActions**()

### uninstallListeners

protected void **uninstallListeners**()

### uninstallDefaults

protected void **uninstallDefaults**()

### createTrackListener

protected [BasicScrollBarUI.TrackListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.TrackListener.html) **createTrackListener**()

### createArrowButtonListener

protected [BasicScrollBarUI.ArrowButtonListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) **createArrowButtonListener**()

### createModelListener

protected [BasicScrollBarUI.ModelListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ModelListener.html) **createModelListener**()

### createScrollListener

protected [BasicScrollBarUI.ScrollListener](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ScrollListener.html) **createScrollListener**()

### createPropertyChangeListener

protected [PropertyChangeListener](http://docs.google.com/java/beans/PropertyChangeListener.html) **createPropertyChangeListener**()

### setThumbRollover

protected void **setThumbRollover**(boolean active)

Sets whether or not the mouse is currently over the thumb.

**Parameters:**active - True indicates the thumb is currently active.**Since:** 1.5

### isThumbRollover

public boolean **isThumbRollover**()

Returns true if the mouse is currently over the thumb.

**Returns:**true if the thumb is currently active**Since:** 1.5

### paint

public void **paint**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)) Paints the specified component appropriate for the look and feel. This method is invoked from the ComponentUI.update method when the specified component is being painted. Subclasses should override this method and use the specified Graphics object to render the content of the component.

**Overrides:**[paint](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**g - the Graphics context in which to paintc - the component being painted; this argument is often ignored, but might be used if the UI object is stateless and shared by multiple components**See Also:**[ComponentUI.update(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(java.awt.Graphics,%20javax.swing.JComponent))

### getPreferredSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getPreferredSize**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

A vertical scrollbar's preferred width is the maximum of preferred widths of the (non null) increment/decrement buttons, and the minimum width of the thumb. The preferred height is the sum of the preferred heights of the same parts. The basis for the preferred size of a horizontal scrollbar is similar.

The preferredSize is only computed once, subsequent calls to this method just return a cached size.

**Overrides:**[getPreferredSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the JScrollBar that's delegating this method to us **Returns:**the preferred size of a Basic JScrollBar**See Also:**[getMaximumSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMaximumSize(javax.swing.JComponent)), [ComponentUI.getMinimumSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent))

### getMaximumSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMaximumSize**([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)

**Description copied from class:** [**ComponentUI**](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) Returns the specified component's maximum size appropriate for the look and feel. If null is returned, the maximum size will be calculated by the component's layout manager instead (this is the preferred approach for any component with a specific layout manager installed). The default implementation of this method invokes getPreferredSize and returns that value.

**Overrides:**[getMaximumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMaximumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - The JScrollBar that's delegating this method to us. **Returns:**new Dimension(Integer.MAX\_VALUE, Integer.MAX\_VALUE);**See Also:**[ComponentUI.getMinimumSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent)), [getPreferredSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getPreferredSize(javax.swing.JComponent))

### createDecreaseButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **createDecreaseButton**(int orientation)

### createIncreaseButton

protected [JButton](http://docs.google.com/javax/swing/JButton.html) **createIncreaseButton**(int orientation)

### paintDecreaseHighlight

protected void **paintDecreaseHighlight**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

### paintIncreaseHighlight

protected void **paintIncreaseHighlight**([Graphics](http://docs.google.com/java/awt/Graphics.html) g)

### paintTrack

protected void **paintTrack**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [JComponent](http://docs.google.com/javax/swing/JComponent.html) c,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) trackBounds)

### paintThumb

protected void **paintThumb**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [JComponent](http://docs.google.com/javax/swing/JComponent.html) c,  
 [Rectangle](http://docs.google.com/java/awt/Rectangle.html) thumbBounds)

### getMinimumThumbSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMinimumThumbSize**()

Return the smallest acceptable size for the thumb. If the scrollbar becomes so small that this size isn't available, the thumb will be hidden.

**Warning** : the value returned by this method should not be be modified, it's a shared static constant.

**Returns:**The smallest acceptable size for the thumb.**See Also:**[getMaximumThumbSize()](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMaximumThumbSize())

### getMaximumThumbSize

protected [Dimension](http://docs.google.com/java/awt/Dimension.html) **getMaximumThumbSize**()

Return the largest acceptable size for the thumb. To create a fixed size thumb one make this method and getMinimumThumbSize return the same value.

**Warning** : the value returned by this method should not be be modified, it's a shared static constant.

**Returns:**The largest acceptable size for the thumb.**See Also:**[getMinimumThumbSize()](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getMinimumThumbSize())

### addLayoutComponent

public void **addLayoutComponent**([String](http://docs.google.com/java/lang/String.html) name,  
 [Component](http://docs.google.com/java/awt/Component.html) child)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#addLayoutComponent(java.lang.String,%20java.awt.Component)) If the layout manager uses a per-component string, adds the component comp to the layout, associating it with the string specified by name.

**Specified by:**[addLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#addLayoutComponent(java.lang.String,%20java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**name - the string to be associated with the componentchild - the component to be added

### removeLayoutComponent

public void **removeLayoutComponent**([Component](http://docs.google.com/java/awt/Component.html) child)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#removeLayoutComponent(java.awt.Component)) Removes the specified component from the layout.

**Specified by:**[removeLayoutComponent](http://docs.google.com/java/awt/LayoutManager.html#removeLayoutComponent(java.awt.Component)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**child - the component to be removed

### preferredLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **preferredLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container)) Calculates the preferred size dimensions for the specified container, given the components it contains.

**Specified by:**[preferredLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**scrollbarContainer - the container to be laid out**See Also:**[LayoutManager.minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container))

### minimumLayoutSize

public [Dimension](http://docs.google.com/java/awt/Dimension.html) **minimumLayoutSize**([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)) Calculates the minimum size dimensions for the specified container, given the components it contains.

**Specified by:**[minimumLayoutSize](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**scrollbarContainer - the component to be laid out**See Also:**[LayoutManager.preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container))

### layoutVScrollbar

protected void **layoutVScrollbar**([JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) sb)

### layoutHScrollbar

protected void **layoutHScrollbar**([JScrollBar](http://docs.google.com/javax/swing/JScrollBar.html) sb)

### layoutContainer

public void **layoutContainer**([Container](http://docs.google.com/java/awt/Container.html) scrollbarContainer)

**Description copied from interface:** [**LayoutManager**](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)) Lays out the specified container.

**Specified by:**[layoutContainer](http://docs.google.com/java/awt/LayoutManager.html#layoutContainer(java.awt.Container)) in interface [LayoutManager](http://docs.google.com/java/awt/LayoutManager.html) **Parameters:**scrollbarContainer - the container to be laid out

### setThumbBounds

protected void **setThumbBounds**(int x,  
 int y,  
 int width,  
 int height)

Set the bounds of the thumb and force a repaint that includes the old thumbBounds and the new one.

**See Also:**[getThumbBounds()](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#getThumbBounds())

### getThumbBounds

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getThumbBounds**()

Return the current size/location of the thumb.

**Warning** : the value returned by this method should not be be modified, it's a reference to the actual rectangle, not a copy.

**Returns:**The current size/location of the thumb.**See Also:**[setThumbBounds(int, int, int, int)](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#setThumbBounds(int,%20int,%20int,%20int))

### getTrackBounds

protected [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **getTrackBounds**()

Returns the current bounds of the track, i.e. the space in between the increment and decrement buttons, less the insets. The value returned by this method is updated each time the scrollbar is laid out (validated).

**Warning** : the value returned by this method should not be be modified, it's a reference to the actual rectangle, not a copy.

**Returns:**the current bounds of the scrollbar track**See Also:**[layoutContainer(java.awt.Container)](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.html#layoutContainer(java.awt.Container))

### scrollByBlock

protected void **scrollByBlock**(int direction)

### scrollByUnit

protected void **scrollByUnit**(int direction)

### getSupportsAbsolutePositioning

public boolean **getSupportsAbsolutePositioning**()

Indicates whether the user can absolutely position the thumb with a mouse gesture (usually the middle mouse button).

**Returns:**true if a mouse gesture can absolutely position the thumb**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicScrollBarUI.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicRootPaneUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicScrollBarUI.ArrowButtonListener.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicScrollBarUI.html)    [**NO FRAMES**](http://docs.google.com/BasicScrollBarUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: [FIELD](#17dp8vu) | [CONSTR](#32hioqz) | [METHOD](#41mghml) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).