| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AsyncBoxView.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/text/AbstractWriter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildLocator.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/AsyncBoxView.html)    [**NO FRAMES**](http://docs.google.com/AsyncBoxView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: [FIELD](#3rdcrjn) | [CONSTR](#lnxbz9) | [METHOD](#1ksv4uv) |

## **javax.swing.text**

Class AsyncBoxView

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [javax.swing.text.View](http://docs.google.com/javax/swing/text/View.html)  
 **javax.swing.text.AsyncBoxView**

**All Implemented Interfaces:** [SwingConstants](http://docs.google.com/javax/swing/SwingConstants.html)

public class **AsyncBoxView**extends [View](http://docs.google.com/javax/swing/text/View.html)

A box that does layout asynchronously. This is useful to keep the GUI event thread moving by not doing any layout on it. The layout is done on a granularity of operations on the child views. After each child view is accessed for some part of layout (a potentially time consuming operation) the remaining tasks can be abandoned or a new higher priority task (i.e. to service a synchronous request or a visible area) can be taken on.

While the child view is being accessed a read lock is aquired on the associated document so that the model is stable while being accessed.

**Since:** 1.3

| **Nested Class Summary** | |
| --- | --- |
| class | [**AsyncBoxView.ChildLocator**](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildLocator.html)            A class to manage the effective position of the child views in a localized area while changes are being made around the localized area. |
| class | [**AsyncBoxView.ChildState**](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html)            A record representing the layout state of a child view. |

| **Field Summary** | |
| --- | --- |
| protected  [AsyncBoxView.ChildLocator](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildLocator.html) | [**locator**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#locator)            Object that manages the offsets of the children. |

| **Fields inherited from class javax.swing.text.**[**View**](http://docs.google.com/javax/swing/text/View.html) |
| --- |
| [BadBreakWeight](http://docs.google.com/javax/swing/text/View.html#BadBreakWeight), [ExcellentBreakWeight](http://docs.google.com/javax/swing/text/View.html#ExcellentBreakWeight), [ForcedBreakWeight](http://docs.google.com/javax/swing/text/View.html#ForcedBreakWeight), [GoodBreakWeight](http://docs.google.com/javax/swing/text/View.html#GoodBreakWeight), [X\_AXIS](http://docs.google.com/javax/swing/text/View.html#X_AXIS), [Y\_AXIS](http://docs.google.com/javax/swing/text/View.html#Y_AXIS) |

| **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** | |
| --- | --- |
| [**AsyncBoxView**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#AsyncBoxView(javax.swing.text.Element,%20int))([Element](http://docs.google.com/javax/swing/text/Element.html) elem, int axis)            Construct a box view that does asynchronous layout. |

| **Method Summary** | |
| --- | --- |
| protected  [AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) | [**createChildState**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#createChildState(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) v)            New ChildState records are created through this method to allow subclasses the extend the ChildState records to do/hold more |
| protected  void | [**flushRequirementChanges**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#flushRequirementChanges())()            Publish the changes in preferences upward to the parent view. |
| float | [**getBottomInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getBottomInset())()            Get the bottom part of the margin around the view. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**getChildAllocation**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getChildAllocation(int,%20java.awt.Shape))(int index, [Shape](http://docs.google.com/java/awt/Shape.html) a)            Fetches the allocation for the given child view. |
| protected  [AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) | [**getChildState**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getChildState(int))(int index)            Fetch the object representing the layout state of of the child at the given index. |
| protected  boolean | [**getEstimatedMajorSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getEstimatedMajorSpan())()            Is the major span currently estimated? |
| protected  float | [**getInsetSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getInsetSpan(int))(int axis)            Fetch the span along an axis that is taken up by the insets. |
| protected  [LayoutQueue](http://docs.google.com/javax/swing/text/LayoutQueue.html) | [**getLayoutQueue**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getLayoutQueue())()            Fetch the queue to use for layout. |
| float | [**getLeftInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getLeftInset())()            Get the left part of the margin around the view. |
| int | [**getMajorAxis**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getMajorAxis())()            Fetch the major axis (the axis the children are tiled along). |
| float | [**getMaximumSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getMaximumSpan(int))(int axis)            Determines the maximum span for this view along an axis. |
| float | [**getMinimumSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getMinimumSpan(int))(int axis)            Determines the minimum span for this view along an axis. |
| int | [**getMinorAxis**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getMinorAxis())()            Fetch the minor axis (the axis orthoginal to the tiled axis). |
| int | [**getNextVisualPositionFrom**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b, [Shape](http://docs.google.com/java/awt/Shape.html) a, int direction, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)            Provides a way to determine the next visually represented model location that one might place a caret. |
| float | [**getPreferredSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getPreferredSpan(int))(int axis)            Determines the preferred span for this view along an axis. |
| float | [**getRightInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getRightInset())()            Get the right part of the margin around the view. |
| float | [**getTopInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getTopInset())()            Get the top part of the margin around the view. |
| [View](http://docs.google.com/javax/swing/text/View.html) | [**getView**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getView(int))(int n)            Gets the nth child view. |
| int | [**getViewCount**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getViewCount())()            Returns the number of views in this view. |
| int | [**getViewIndex**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getViewIndex(int,%20javax.swing.text.Position.Bias))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)            Returns the child view index representing the given position in the model. |
| protected  int | [**getViewIndexAtPosition**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#getViewIndexAtPosition(int,%20javax.swing.text.Position.Bias))(int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)            Fetches the child view index representing the given position in the model. |
| protected  void | [**loadChildren**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#loadChildren(javax.swing.text.ViewFactory))([ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            Loads all of the children to initialize the view. |
| protected  void | [**majorRequirementChange**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#majorRequirementChange(javax.swing.text.AsyncBoxView.ChildState,%20float))([AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) cs, float delta)            Requirements changed along the major axis. |
| protected  void | [**minorRequirementChange**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#minorRequirementChange(javax.swing.text.AsyncBoxView.ChildState))([AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) cs)            Requirements changed along the minor axis. |
| [Shape](http://docs.google.com/java/awt/Shape.html) | [**modelToView**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias))(int pos, [Shape](http://docs.google.com/java/awt/Shape.html) a, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)            Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it. |
| void | [**paint**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#paint(java.awt.Graphics,%20java.awt.Shape))([Graphics](http://docs.google.com/java/awt/Graphics.html) g, [Shape](http://docs.google.com/java/awt/Shape.html) alloc)            Render the view using the given allocation and rendering surface. |
| void | [**preferenceChanged**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean))([View](http://docs.google.com/javax/swing/text/View.html) child, boolean width, boolean height)            Child views can call this on the parent to indicate that the preference has changed and should be reconsidered for layout. |
| void | [**replace**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D))(int offset, int length, [View](http://docs.google.com/javax/swing/text/View.html)[] views)            Calls the superclass to update the child views, and updates the status records for the children. |
| void | [**setBottomInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setBottomInset(float))(float i)            Set the bottom part of the margin around the view. |
| protected  void | [**setEstimatedMajorSpan**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setEstimatedMajorSpan(boolean))(boolean isEstimated)            Set the estimatedMajorSpan property that determines if the major span should be treated as being estimated. |
| void | [**setLeftInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setLeftInset(float))(float i)            Set the left part of the margin around the view. |
| void | [**setParent**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setParent(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) parent)            Sets the parent of the view. |
| void | [**setRightInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setRightInset(float))(float i)            Set the right part of the margin around the view. |
| void | [**setSize**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setSize(float,%20float))(float width, float height)            Sets the size of the view. |
| void | [**setTopInset**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setTopInset(float))(float i)            Set the top part of the margin around the view. |
| protected  void | [**updateLayout**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#updateLayout(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape))([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec, [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e, [Shape](http://docs.google.com/java/awt/Shape.html) a)            Update the layout in response to receiving notification of change from the model. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/text/AsyncBoxView.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D))(float x, float y, [Shape](http://docs.google.com/java/awt/Shape.html) a, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasReturn)            Provides a mapping from the view coordinate space to the logical coordinate space of the model. |

| **Methods inherited from class javax.swing.text.**[**View**](http://docs.google.com/javax/swing/text/View.html) |
| --- |
| [append](http://docs.google.com/javax/swing/text/View.html#append(javax.swing.text.View)), [breakView](http://docs.google.com/javax/swing/text/View.html#breakView(int,%20int,%20float,%20float)), [changedUpdate](http://docs.google.com/javax/swing/text/View.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [createFragment](http://docs.google.com/javax/swing/text/View.html#createFragment(int,%20int)), [forwardUpdate](http://docs.google.com/javax/swing/text/View.html#forwardUpdate(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [forwardUpdateToView](http://docs.google.com/javax/swing/text/View.html#forwardUpdateToView(javax.swing.text.View,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [getAlignment](http://docs.google.com/javax/swing/text/View.html#getAlignment(int)), [getAttributes](http://docs.google.com/javax/swing/text/View.html#getAttributes()), [getBreakWeight](http://docs.google.com/javax/swing/text/View.html#getBreakWeight(int,%20float,%20float)), [getContainer](http://docs.google.com/javax/swing/text/View.html#getContainer()), [getDocument](http://docs.google.com/javax/swing/text/View.html#getDocument()), [getElement](http://docs.google.com/javax/swing/text/View.html#getElement()), [getEndOffset](http://docs.google.com/javax/swing/text/View.html#getEndOffset()), [getGraphics](http://docs.google.com/javax/swing/text/View.html#getGraphics()), [getParent](http://docs.google.com/javax/swing/text/View.html#getParent()), [getResizeWeight](http://docs.google.com/javax/swing/text/View.html#getResizeWeight(int)), [getStartOffset](http://docs.google.com/javax/swing/text/View.html#getStartOffset()), [getToolTipText](http://docs.google.com/javax/swing/text/View.html#getToolTipText(float,%20float,%20java.awt.Shape)), [getViewFactory](http://docs.google.com/javax/swing/text/View.html#getViewFactory()), [getViewIndex](http://docs.google.com/javax/swing/text/View.html#getViewIndex(float,%20float,%20java.awt.Shape)), [insert](http://docs.google.com/javax/swing/text/View.html#insert(int,%20javax.swing.text.View)), [insertUpdate](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [isVisible](http://docs.google.com/javax/swing/text/View.html#isVisible()), [modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias,%20java.awt.Shape)), [modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20java.awt.Shape)), [remove](http://docs.google.com/javax/swing/text/View.html#remove(int)), [removeAll](http://docs.google.com/javax/swing/text/View.html#removeAll()), [removeUpdate](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [updateChildren](http://docs.google.com/javax/swing/text/View.html#updateChildren(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20javax.swing.text.ViewFactory)), [viewToModel](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape)) |

| **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** |
| --- |

### locator

protected [AsyncBoxView.ChildLocator](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildLocator.html) **locator**

Object that manages the offsets of the children. All locking for management of child locations is on this object.

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

### AsyncBoxView

public **AsyncBoxView**([Element](http://docs.google.com/javax/swing/text/Element.html) elem,  
 int axis)

Construct a box view that does asynchronous layout.

**Parameters:**elem - the element of the model to representaxis - the axis to tile along. This can be either X\_AXIS or Y\_AXIS.

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

### getMajorAxis

public int **getMajorAxis**()

Fetch the major axis (the axis the children are tiled along). This will have a value of either X\_AXIS or Y\_AXIS.

### getMinorAxis

public int **getMinorAxis**()

Fetch the minor axis (the axis orthoginal to the tiled axis). This will have a value of either X\_AXIS or Y\_AXIS.

### getTopInset

public float **getTopInset**()

Get the top part of the margin around the view.

### setTopInset

public void **setTopInset**(float i)

Set the top part of the margin around the view.

**Parameters:**i - the value of the inset

### getBottomInset

public float **getBottomInset**()

Get the bottom part of the margin around the view.

### setBottomInset

public void **setBottomInset**(float i)

Set the bottom part of the margin around the view.

**Parameters:**i - the value of the inset

### getLeftInset

public float **getLeftInset**()

Get the left part of the margin around the view.

### setLeftInset

public void **setLeftInset**(float i)

Set the left part of the margin around the view.

**Parameters:**i - the value of the inset

### getRightInset

public float **getRightInset**()

Get the right part of the margin around the view.

### setRightInset

public void **setRightInset**(float i)

Set the right part of the margin around the view.

**Parameters:**i - the value of the inset

### getInsetSpan

protected float **getInsetSpan**(int axis)

Fetch the span along an axis that is taken up by the insets.

**Parameters:**axis - the axis to determine the total insets along, either X\_AXIS or Y\_AXIS.**Since:** 1.4

### setEstimatedMajorSpan

protected void **setEstimatedMajorSpan**(boolean isEstimated)

Set the estimatedMajorSpan property that determines if the major span should be treated as being estimated. If this property is true, the value of setSize along the major axis will change the requirements along the major axis and incremental changes will be ignored until all of the children have been updated (which will cause the property to automatically be set to false). If the property is false the value of the majorSpan will be considered to be accurate and incremental changes will be added into the total as they are calculated.

**Since:** 1.4

### getEstimatedMajorSpan

protected boolean **getEstimatedMajorSpan**()

Is the major span currently estimated?

**Since:** 1.4

### getChildState

protected [AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) **getChildState**(int index)

Fetch the object representing the layout state of of the child at the given index.

**Parameters:**index - the child index. This should be a value >= 0 and < getViewCount().

### getLayoutQueue

protected [LayoutQueue](http://docs.google.com/javax/swing/text/LayoutQueue.html) **getLayoutQueue**()

Fetch the queue to use for layout.

### createChildState

protected [AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) **createChildState**([View](http://docs.google.com/javax/swing/text/View.html) v)

New ChildState records are created through this method to allow subclasses the extend the ChildState records to do/hold more

### majorRequirementChange

protected void **majorRequirementChange**([AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) cs,  
 float delta)

Requirements changed along the major axis. This is called by the thread doing layout for the given ChildState object when it has completed fetching the child views new preferences. Typically this would be the layout thread, but might be the event thread if it is trying to update something immediately (such as to perform a model/view translation).

This is implemented to mark the major axis as having changed so that a future check to see if the requirements need to be published to the parent view will consider the major axis. If the span along the major axis is not estimated, it is updated by the given delta to reflect the incremental change. The delta is ignored if the major span is estimated.

### minorRequirementChange

protected void **minorRequirementChange**([AsyncBoxView.ChildState](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildState.html) cs)

Requirements changed along the minor axis. This is called by the thread doing layout for the given ChildState object when it has completed fetching the child views new preferences. Typically this would be the layout thread, but might be the GUI thread if it is trying to update something immediately (such as to perform a model/view translation).

### flushRequirementChanges

protected void **flushRequirementChanges**()

Publish the changes in preferences upward to the parent view. This is normally called by the layout thread.

### replace

public void **replace**(int offset,  
 int length,  
 [View](http://docs.google.com/javax/swing/text/View.html)[] views)

Calls the superclass to update the child views, and updates the status records for the children. This is expected to be called while a write lock is held on the model so that interaction with the layout thread will not happen (i.e. the layout thread acquires a read lock before doing anything).

**Overrides:**[replace](http://docs.google.com/javax/swing/text/View.html#replace(int,%20int,%20javax.swing.text.View%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**offset - the starting offset into the child views >= 0length - the number of existing views to replace >= 0views - the child views to insert

### loadChildren

protected void **loadChildren**([ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

Loads all of the children to initialize the view. This is called by the [setParent](#2zbgiuw) method. Subclasses can reimplement this to initialize their child views in a different manner. The default implementation creates a child view for each child element.

Normally a write-lock is held on the Document while the children are being changed, which keeps the rendering and layout threads safe. The exception to this is when the view is initialized to represent an existing element (via this method), so it is synchronized to exclude preferenceChanged while we are initializing.

**Parameters:**f - the view factory**See Also:**[setParent(javax.swing.text.View)](http://docs.google.com/javax/swing/text/AsyncBoxView.html#setParent(javax.swing.text.View))

### getViewIndexAtPosition

protected int **getViewIndexAtPosition**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)

Fetches the child view index representing the given position in the model. This is implemented to fetch the view in the case where there is a child view for each child element.

**Parameters:**pos - the position >= 0 **Returns:**index of the view representing the given position, or -1 if no view represents that position

### updateLayout

protected void **updateLayout**([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec,  
 [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a)

Update the layout in response to receiving notification of change from the model. This is implemented to note the change on the ChildLocator so that offsets of the children will be correctly computed.

**Overrides:**[updateLayout](http://docs.google.com/javax/swing/text/View.html#updateLayout(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**ec - changes to the element this view is responsible for (may be null if there were no changes).e - the change information from the associated documenta - the current allocation of the view**See Also:**[View.insertUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [View.removeUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [View.changedUpdate(javax.swing.event.DocumentEvent, java.awt.Shape, javax.swing.text.ViewFactory)](http://docs.google.com/javax/swing/text/View.html#changedUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))

### setParent

public void **setParent**([View](http://docs.google.com/javax/swing/text/View.html) parent)

Sets the parent of the view. This is reimplemented to provide the superclass behavior as well as calling the loadChildren method if this view does not already have children. The children should not be loaded in the constructor because the act of setting the parent may cause them to try to search up the hierarchy (to get the hosting Container for example). If this view has children (the view is being moved from one place in the view hierarchy to another), the loadChildren method will not be called.

**Overrides:**[setParent](http://docs.google.com/javax/swing/text/View.html#setParent(javax.swing.text.View)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**parent - the parent of the view, null if none

### preferenceChanged

public void **preferenceChanged**([View](http://docs.google.com/javax/swing/text/View.html) child,  
 boolean width,  
 boolean height)

Child views can call this on the parent to indicate that the preference has changed and should be reconsidered for layout. This is reimplemented to queue new work on the layout thread. This method gets messaged from multiple threads via the children.

**Overrides:**[preferenceChanged](http://docs.google.com/javax/swing/text/View.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**child - the child viewwidth - true if the width preference has changedheight - true if the height preference has changed**See Also:**[JComponent.revalidate()](http://docs.google.com/javax/swing/JComponent.html#revalidate())

### setSize

public void **setSize**(float width,  
 float height)

Sets the size of the view. This should cause layout of the view if the view caches any layout information.

Since the major axis is updated asynchronously and should be the sum of the tiled children the call is ignored for the major axis. Since the minor axis is flexible, work is queued to resize the children if the minor span changes.

**Overrides:**[setSize](http://docs.google.com/javax/swing/text/View.html#setSize(float,%20float)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**width - the width >= 0height - the height >= 0

### paint

public void **paint**([Graphics](http://docs.google.com/java/awt/Graphics.html) g,  
 [Shape](http://docs.google.com/java/awt/Shape.html) alloc)

Render the view using the given allocation and rendering surface.

This is implemented to determine whether or not the desired region to be rendered (i.e. the unclipped area) is up to date or not. If up-to-date the children are rendered. If not up-to-date, a task to build the desired area is placed on the layout queue as a high priority task. This keeps by event thread moving by rendering if ready, and postponing until a later time if not ready (since paint requests can be rescheduled).

**Specified by:**[paint](http://docs.google.com/javax/swing/text/View.html#paint(java.awt.Graphics,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**g - the rendering surface to usealloc - the allocated region to render into**See Also:**[View.paint(java.awt.Graphics, java.awt.Shape)](http://docs.google.com/javax/swing/text/View.html#paint(java.awt.Graphics,%20java.awt.Shape))

### getPreferredSpan

public float **getPreferredSpan**(int axis)

Determines the preferred span for this view along an axis.

**Specified by:**[getPreferredSpan](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0. Typically the view is told to render into the span that is returned, although there is no guarantee. The parent may choose to resize or break the view. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### getMinimumSpan

public float **getMinimumSpan**(int axis)

Determines the minimum span for this view along an axis.

**Overrides:**[getMinimumSpan](http://docs.google.com/javax/swing/text/View.html#getMinimumSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0. Typically the view is told to render into the span that is returned, although there is no guarantee. The parent may choose to resize or break the view. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### getMaximumSpan

public float **getMaximumSpan**(int axis)

Determines the maximum span for this view along an axis.

**Overrides:**[getMaximumSpan](http://docs.google.com/javax/swing/text/View.html#getMaximumSpan(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**axis - may be either View.X\_AXIS or View.Y\_AXIS **Returns:**the span the view would like to be rendered into >= 0. Typically the view is told to render into the span that is returned, although there is no guarantee. The parent may choose to resize or break the view. **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid axis type**See Also:**[View.getPreferredSpan(int)](http://docs.google.com/javax/swing/text/View.html#getPreferredSpan(int))

### getViewCount

public int **getViewCount**()

Returns the number of views in this view. Since the default is to not be a composite view this returns 0.

**Overrides:**[getViewCount](http://docs.google.com/javax/swing/text/View.html#getViewCount()) in class [View](http://docs.google.com/javax/swing/text/View.html) **Returns:**the number of views >= 0**See Also:**[View.getViewCount()](http://docs.google.com/javax/swing/text/View.html#getViewCount())

### getView

public [View](http://docs.google.com/javax/swing/text/View.html) **getView**(int n)

Gets the nth child view. Since there are no children by default, this returns null.

**Overrides:**[getView](http://docs.google.com/javax/swing/text/View.html#getView(int)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**n - the number of the view to get, >= 0 && < getViewCount() **Returns:**the view

### getChildAllocation

public [Shape](http://docs.google.com/java/awt/Shape.html) **getChildAllocation**(int index,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a)

Fetches the allocation for the given child view. This enables finding out where various views are located, without assuming the views store their location. This returns null since the default is to not have any child views.

**Overrides:**[getChildAllocation](http://docs.google.com/javax/swing/text/View.html#getChildAllocation(int,%20java.awt.Shape)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**index - the index of the child, >= 0 && < getViewCount()a - the allocation to this view. **Returns:**the allocation to the child

### getViewIndex

public int **getViewIndex**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)

Returns the child view index representing the given position in the model. By default a view has no children so this is implemented to return -1 to indicate there is no valid child index for any position.

**Overrides:**[getViewIndex](http://docs.google.com/javax/swing/text/View.html#getViewIndex(int,%20javax.swing.text.Position.Bias)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position >= 0 **Returns:**index of the view representing the given position, or -1 if no view represents that position**Since:** 1.3

### modelToView

public [Shape](http://docs.google.com/java/awt/Shape.html) **modelToView**(int pos,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Provides a mapping from the document model coordinate space to the coordinate space of the view mapped to it.

**Specified by:**[modelToView](http://docs.google.com/javax/swing/text/View.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position to convert >= 0a - the allocated region to render intob - the bias toward the previous character or the next character represented by the offset, in case the position is a boundary of two views. **Returns:**the bounding box of the given position is returned **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) - if the given position does not represent a valid location in the associated document [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - for an invalid bias argument**See Also:**[View.viewToModel(float, float, java.awt.Shape, javax.swing.text.Position.Bias[])](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D))

### viewToModel

public int **viewToModel**(float x,  
 float y,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasReturn)

Provides a mapping from the view coordinate space to the logical coordinate space of the model. The biasReturn argument will be filled in to indicate that the point given is closer to the next character in the model or the previous character in the model.

This is expected to be called by the GUI thread, holding a read-lock on the associated model. It is implemented to locate the child view and determine it's allocation with a lock on the ChildLocator object, and to call viewToModel on the child view with a lock on the ChildState object to avoid interaction with the layout thread.

**Specified by:**[viewToModel](http://docs.google.com/javax/swing/text/View.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**x - the X coordinate >= 0y - the Y coordinate >= 0a - the allocated region to render into **Returns:**the location within the model that best represents the given point in the view >= 0. The biasReturn argument will be filled in to indicate that the point given is closer to the next character in the model or the previous character in the model.

### getNextVisualPositionFrom

public int **getNextVisualPositionFrom**(int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 int direction,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasRet)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Provides a way to determine the next visually represented model location that one might place a caret. Some views may not be visible, they might not be in the same order found in the model, or they just might not allow access to some of the locations in the model.

**Overrides:**[getNextVisualPositionFrom](http://docs.google.com/javax/swing/text/View.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**pos - the position to convert >= 0a - the allocated region to render intodirection - the direction from the current position that can be thought of as the arrow keys typically found on a keyboard; this may be one of the following:

SwingConstants.WEST SwingConstants.EAST SwingConstants.NORTH SwingConstants.SOUTHbiasRet - an array contain the bias that was checked **Returns:**the location within the model that best represents the next location visual position **Throws:** [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html) [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if direction is invalid

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AsyncBoxView.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/text/AbstractWriter.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/text/AsyncBoxView.ChildLocator.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/AsyncBoxView.html)    [**NO FRAMES**](http://docs.google.com/AsyncBoxView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | [FIELD](#2et92p0) | [CONSTR](#1t3h5sf) | [METHOD](#4d34og8) | DETAIL: [FIELD](#3rdcrjn) | [CONSTR](#lnxbz9) | [METHOD](#1ksv4uv) |

[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).