| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ZoneView.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/WrappedPlainView.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/ZoneView.html)    [**NO FRAMES**](http://docs.google.com/ZoneView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#35nkun2) |

## **javax.swing.text**

Class ZoneView

[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.CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html)  
 [javax.swing.text.BoxView](http://docs.google.com/javax/swing/text/BoxView.html)  
 **javax.swing.text.ZoneView**

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

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

ZoneView is a View implementation that creates zones for which the child views are not created or stored until they are needed for display or model/view translations. This enables a substantial reduction in memory consumption for situations where the model being represented is very large, by building view objects only for the region being actively viewed/edited. The size of the children can be estimated in some way, or calculated asynchronously with only the result being saved.

ZoneView extends BoxView to provide a box that implements zones for its children. The zones are special View implementations (the children of an instance of this class) that represent only a portion of the model that an instance of ZoneView is responsible for. The zones don't create child views until an attempt is made to display them. A box shaped view is well suited to this because:

* Boxes are a heavily used view, and having a box that provides this behavior gives substantial opportunity to plug the behavior into a view hierarchy from the view factory.
* Boxes are tiled in one direction, so it is easy to divide them into zones in a reliable way.
* Boxes typically have a simple relationship to the model (i.e. they create child views that directly represent the child elements).
* Boxes are easier to estimate the size of than some other shapes.

The default behavior is controled by two properties, maxZoneSize and maxZonesLoaded. Setting maxZoneSize to Integer.MAX\_VALUE would have the effect of causing only one zone to be created. This would effectively turn the view into an implementation of the decorator pattern. Setting maxZonesLoaded to a value of Integer.MAX\_VALUE would cause zones to never be unloaded. For simplicity, zones are created on boundaries represented by the child elements of the element the view is responsible for. The zones can be any View implementation, but the default implementation is based upon AsyncBoxView which supports fairly large zones efficiently.

**Since:** 1.3 **See Also:**[View](http://docs.google.com/javax/swing/text/View.html)

| **Field Summary** | |
| --- | --- |

| **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** | |
| --- | --- |
| [**ZoneView**](http://docs.google.com/javax/swing/text/ZoneView.html#ZoneView(javax.swing.text.Element,%20int))([Element](http://docs.google.com/javax/swing/text/Element.html) elem, int axis)            Constructs a ZoneView. |

| **Method Summary** | |
| --- | --- |
| protected  [View](http://docs.google.com/javax/swing/text/View.html) | [**createZone**](http://docs.google.com/javax/swing/text/ZoneView.html#createZone(int,%20int))(int p0, int p1)            Create a view to represent a zone for the given range within the model (which should be within the range of this objects responsibility). |
| int | [**getMaximumZoneSize**](http://docs.google.com/javax/swing/text/ZoneView.html#getMaximumZoneSize())()            Get the current maximum zone size. |
| int | [**getMaxZonesLoaded**](http://docs.google.com/javax/swing/text/ZoneView.html#getMaxZonesLoaded())()            Get the current setting of the number of zones allowed to be loaded at the same time. |
| protected  int | [**getViewIndexAtPosition**](http://docs.google.com/javax/swing/text/ZoneView.html#getViewIndexAtPosition(int))(int pos)            Returns the child view index representing the given position in the model. |
| void | [**insertUpdate**](http://docs.google.com/javax/swing/text/ZoneView.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) changes, [Shape](http://docs.google.com/java/awt/Shape.html) a, [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            Gives notification that something was inserted into the document in a location that this view is responsible for. |
| protected  boolean | [**isZoneLoaded**](http://docs.google.com/javax/swing/text/ZoneView.html#isZoneLoaded(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) zone)            Determine if a zone is in the loaded state. |
| protected  void | [**loadChildren**](http://docs.google.com/javax/swing/text/ZoneView.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. |
| void | [**removeUpdate**](http://docs.google.com/javax/swing/text/ZoneView.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory))([DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) changes, [Shape](http://docs.google.com/java/awt/Shape.html) a, [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            Gives notification that something was removed from the document in a location that this view is responsible for. |
| void | [**setMaximumZoneSize**](http://docs.google.com/javax/swing/text/ZoneView.html#setMaximumZoneSize(int))(int size)            Set the desired maximum zone size. |
| void | [**setMaxZonesLoaded**](http://docs.google.com/javax/swing/text/ZoneView.html#setMaxZonesLoaded(int))(int mzl)            Sets the current setting of the number of zones allowed to be loaded at the same time. |
| protected  void | [**unloadZone**](http://docs.google.com/javax/swing/text/ZoneView.html#unloadZone(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) zone)            Unload a zone (Convert the zone to its memory saving state). |
| protected  boolean | [**updateChildren**](http://docs.google.com/javax/swing/text/ZoneView.html#updateChildren(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20javax.swing.text.ViewFactory))([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec, [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e, [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)            The superclass behavior will try to update the child views which is not desired in this case, since the children are zones and not directly effected by the changes to the associated element. |
| protected  void | [**zoneWasLoaded**](http://docs.google.com/javax/swing/text/ZoneView.html#zoneWasLoaded(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) zone)            Called by a zone when it gets loaded. |

| **Methods inherited from class javax.swing.text.**[**BoxView**](http://docs.google.com/javax/swing/text/BoxView.html) |
| --- |
| [baselineLayout](http://docs.google.com/javax/swing/text/BoxView.html#baselineLayout(int,%20int,%20int%5B%5D,%20int%5B%5D)), [baselineRequirements](http://docs.google.com/javax/swing/text/BoxView.html#baselineRequirements(int,%20javax.swing.SizeRequirements)), [calculateMajorAxisRequirements](http://docs.google.com/javax/swing/text/BoxView.html#calculateMajorAxisRequirements(int,%20javax.swing.SizeRequirements)), [calculateMinorAxisRequirements](http://docs.google.com/javax/swing/text/BoxView.html#calculateMinorAxisRequirements(int,%20javax.swing.SizeRequirements)), [childAllocation](http://docs.google.com/javax/swing/text/BoxView.html#childAllocation(int,%20java.awt.Rectangle)), [flipEastAndWestAtEnds](http://docs.google.com/javax/swing/text/BoxView.html#flipEastAndWestAtEnds(int,%20javax.swing.text.Position.Bias)), [forwardUpdate](http://docs.google.com/javax/swing/text/BoxView.html#forwardUpdate(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)), [getAlignment](http://docs.google.com/javax/swing/text/BoxView.html#getAlignment(int)), [getAxis](http://docs.google.com/javax/swing/text/BoxView.html#getAxis()), [getChildAllocation](http://docs.google.com/javax/swing/text/BoxView.html#getChildAllocation(int,%20java.awt.Shape)), [getHeight](http://docs.google.com/javax/swing/text/BoxView.html#getHeight()), [getMaximumSpan](http://docs.google.com/javax/swing/text/BoxView.html#getMaximumSpan(int)), [getMinimumSpan](http://docs.google.com/javax/swing/text/BoxView.html#getMinimumSpan(int)), [getOffset](http://docs.google.com/javax/swing/text/BoxView.html#getOffset(int,%20int)), [getPreferredSpan](http://docs.google.com/javax/swing/text/BoxView.html#getPreferredSpan(int)), [getResizeWeight](http://docs.google.com/javax/swing/text/BoxView.html#getResizeWeight(int)), [getSpan](http://docs.google.com/javax/swing/text/BoxView.html#getSpan(int,%20int)), [getViewAtPoint](http://docs.google.com/javax/swing/text/BoxView.html#getViewAtPoint(int,%20int,%20java.awt.Rectangle)), [getWidth](http://docs.google.com/javax/swing/text/BoxView.html#getWidth()), [isAfter](http://docs.google.com/javax/swing/text/BoxView.html#isAfter(int,%20int,%20java.awt.Rectangle)), [isAllocationValid](http://docs.google.com/javax/swing/text/BoxView.html#isAllocationValid()), [isBefore](http://docs.google.com/javax/swing/text/BoxView.html#isBefore(int,%20int,%20java.awt.Rectangle)), [isLayoutValid](http://docs.google.com/javax/swing/text/BoxView.html#isLayoutValid(int)), [layout](http://docs.google.com/javax/swing/text/BoxView.html#layout(int,%20int)), [layoutChanged](http://docs.google.com/javax/swing/text/BoxView.html#layoutChanged(int)), [layoutMajorAxis](http://docs.google.com/javax/swing/text/BoxView.html#layoutMajorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D)), [layoutMinorAxis](http://docs.google.com/javax/swing/text/BoxView.html#layoutMinorAxis(int,%20int,%20int%5B%5D,%20int%5B%5D)), [modelToView](http://docs.google.com/javax/swing/text/BoxView.html#modelToView(int,%20java.awt.Shape,%20javax.swing.text.Position.Bias)), [paint](http://docs.google.com/javax/swing/text/BoxView.html#paint(java.awt.Graphics,%20java.awt.Shape)), [paintChild](http://docs.google.com/javax/swing/text/BoxView.html#paintChild(java.awt.Graphics,%20java.awt.Rectangle,%20int)), [preferenceChanged](http://docs.google.com/javax/swing/text/BoxView.html#preferenceChanged(javax.swing.text.View,%20boolean,%20boolean)), [replace](http://docs.google.com/javax/swing/text/BoxView.html#replace(int,%20int,%20javax.swing.text.View%5B%5D)), [setAxis](http://docs.google.com/javax/swing/text/BoxView.html#setAxis(int)), [setSize](http://docs.google.com/javax/swing/text/BoxView.html#setSize(float,%20float)), [viewToModel](http://docs.google.com/javax/swing/text/BoxView.html#viewToModel(float,%20float,%20java.awt.Shape,%20javax.swing.text.Position.Bias%5B%5D)) |

| **Methods inherited from class javax.swing.text.**[**CompositeView**](http://docs.google.com/javax/swing/text/CompositeView.html) |
| --- |
| [getBottomInset](http://docs.google.com/javax/swing/text/CompositeView.html#getBottomInset()), [getInsideAllocation](http://docs.google.com/javax/swing/text/CompositeView.html#getInsideAllocation(java.awt.Shape)), [getLeftInset](http://docs.google.com/javax/swing/text/CompositeView.html#getLeftInset()), [getNextEastWestVisualPositionFrom](http://docs.google.com/javax/swing/text/CompositeView.html#getNextEastWestVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)), [getNextNorthSouthVisualPositionFrom](http://docs.google.com/javax/swing/text/CompositeView.html#getNextNorthSouthVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)), [getNextVisualPositionFrom](http://docs.google.com/javax/swing/text/CompositeView.html#getNextVisualPositionFrom(int,%20javax.swing.text.Position.Bias,%20java.awt.Shape,%20int,%20javax.swing.text.Position.Bias%5B%5D)), [getRightInset](http://docs.google.com/javax/swing/text/CompositeView.html#getRightInset()), [getTopInset](http://docs.google.com/javax/swing/text/CompositeView.html#getTopInset()), [getView](http://docs.google.com/javax/swing/text/CompositeView.html#getView(int)), [getViewAtPosition](http://docs.google.com/javax/swing/text/CompositeView.html#getViewAtPosition(int,%20java.awt.Rectangle)), [getViewCount](http://docs.google.com/javax/swing/text/CompositeView.html#getViewCount()), [getViewIndex](http://docs.google.com/javax/swing/text/CompositeView.html#getViewIndex(int,%20javax.swing.text.Position.Bias)), [modelToView](http://docs.google.com/javax/swing/text/CompositeView.html#modelToView(int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias,%20java.awt.Shape)), [setInsets](http://docs.google.com/javax/swing/text/CompositeView.html#setInsets(short,%20short,%20short,%20short)), [setParagraphInsets](http://docs.google.com/javax/swing/text/CompositeView.html#setParagraphInsets(javax.swing.text.AttributeSet)), [setParent](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View)) |

| **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)), [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)), [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()), [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)), [isVisible](http://docs.google.com/javax/swing/text/View.html#isVisible()), [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()), [updateLayout](http://docs.google.com/javax/swing/text/View.html#updateLayout(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20java.awt.Shape)), [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)) |

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

### ZoneView

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

Constructs a ZoneView.

**Parameters:**elem - the element this view is responsible foraxis - either View.X\_AXIS or View.Y\_AXIS

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

### getMaximumZoneSize

public int **getMaximumZoneSize**()

Get the current maximum zone size.

### setMaximumZoneSize

public void **setMaximumZoneSize**(int size)

Set the desired maximum zone size. A zone may get larger than this size if a single child view is larger than this size since zones are formed on child view boundaries.

**Parameters:**size - the number of characters the zone may represent before attempting to break the zone into a smaller size.

### getMaxZonesLoaded

public int **getMaxZonesLoaded**()

Get the current setting of the number of zones allowed to be loaded at the same time.

### setMaxZonesLoaded

public void **setMaxZonesLoaded**(int mzl)

Sets the current setting of the number of zones allowed to be loaded at the same time. This will throw an IllegalArgumentException if mzl is less than 1.

**Parameters:**mzl - the desired maximum number of zones to be actively loaded, must be greater than 0 **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if mzl is < 1

### zoneWasLoaded

protected void **zoneWasLoaded**([View](http://docs.google.com/javax/swing/text/View.html) zone)

Called by a zone when it gets loaded. This happens when an attempt is made to display or perform a model/view translation on a zone that was in an unloaded state. This is imlemented to check if the maximum number of zones was reached and to unload the oldest zone if so.

**Parameters:**zone - the child view that was just loaded.

### unloadZone

protected void **unloadZone**([View](http://docs.google.com/javax/swing/text/View.html) zone)

Unload a zone (Convert the zone to its memory saving state). The zones are expected to represent a subset of the child elements of the element this view is responsible for. Therefore, the default implementation is to simple remove all the children.

**Parameters:**zone - the child view desired to be set to an unloaded state.

### isZoneLoaded

protected boolean **isZoneLoaded**([View](http://docs.google.com/javax/swing/text/View.html) zone)

Determine if a zone is in the loaded state. The zones are expected to represent a subset of the child elements of the element this view is responsible for. Therefore, the default implementation is to return true if the view has children.

### createZone

protected [View](http://docs.google.com/javax/swing/text/View.html) **createZone**(int p0,  
 int p1)

Create a view to represent a zone for the given range within the model (which should be within the range of this objects responsibility). This is called by the zone management logic to create new zones. Subclasses can provide a different implementation for a zone by changing this method.

**Parameters:**p0 - the start of the desired zone. This should be >= getStartOffset() and < getEndOffset(). This value should also be < p1.p1 - the end of the desired zone. This should be > getStartOffset() and <= getEndOffset(). This value should also be > p0.

### 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 method. This is reimplemented to not load any children directly (as they are created by the zones). This method creates the initial set of zones. Zones don't actually get populated however until an attempt is made to display them or to do model/view coordinate translation.

**Overrides:**[loadChildren](http://docs.google.com/javax/swing/text/CompositeView.html#loadChildren(javax.swing.text.ViewFactory)) in class [CompositeView](http://docs.google.com/javax/swing/text/CompositeView.html) **Parameters:**f - the view factory**See Also:**[CompositeView.setParent(javax.swing.text.View)](http://docs.google.com/javax/swing/text/CompositeView.html#setParent(javax.swing.text.View))

### getViewIndexAtPosition

protected int **getViewIndexAtPosition**(int pos)

Returns the child view index representing the given position in the model.

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

### updateChildren

protected boolean **updateChildren**([DocumentEvent.ElementChange](http://docs.google.com/javax/swing/event/DocumentEvent.ElementChange.html) ec,  
 [DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) e,  
 [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

The superclass behavior will try to update the child views which is not desired in this case, since the children are zones and not directly effected by the changes to the associated element. This is reimplemented to do nothing and return false.

**Overrides:**[updateChildren](http://docs.google.com/javax/swing/text/View.html#updateChildren(javax.swing.event.DocumentEvent.ElementChange,%20javax.swing.event.DocumentEvent,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**ec - the change information for the element this view is responsible for. This should not be null if this method gets callede - the change information from the associated documentf - the factory to use to build child views **Returns:**whether or not the child views represent the child elements of the element this view is responsible for. Some views create children that represent a portion of the element they are responsible for, and should return false. This information is used to determine if views in the range of the added elements should be forwarded to or not**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))

### insertUpdate

public void **insertUpdate**([DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) changes,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

Gives notification that something was inserted into the document in a location that this view is responsible for. This is largely delegated to the superclass, but is reimplemented to update the relevant zone (i.e. determine if a zone needs to be split into a set of 2 or more zones).

**Overrides:**[insertUpdate](http://docs.google.com/javax/swing/text/View.html#insertUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**changes - the change information from the associated documenta - the current allocation of the viewf - the factory to use to rebuild if the view has children**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))

### removeUpdate

public void **removeUpdate**([DocumentEvent](http://docs.google.com/javax/swing/event/DocumentEvent.html) changes,  
 [Shape](http://docs.google.com/java/awt/Shape.html) a,  
 [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) f)

Gives notification that something was removed from the document in a location that this view is responsible for. This is largely delegated to the superclass, but is reimplemented to update the relevant zones (i.e. determine if zones need to be removed or joined with another zone).

**Overrides:**[removeUpdate](http://docs.google.com/javax/swing/text/View.html#removeUpdate(javax.swing.event.DocumentEvent,%20java.awt.Shape,%20javax.swing.text.ViewFactory)) in class [View](http://docs.google.com/javax/swing/text/View.html) **Parameters:**changes - the change information from the associated documenta - the current allocation of the viewf - the factory to use to rebuild if the view has children**See Also:**[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))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/ZoneView.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/WrappedPlainView.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?javax/swing/text/ZoneView.html)    [**NO FRAMES**](http://docs.google.com/ZoneView.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#2et92p0) | [CONSTR](#3dy6vkm) | [METHOD](#1t3h5sf) | DETAIL: FIELD | [CONSTR](#26in1rg) | [METHOD](#35nkun2) |

[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).