| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicTextUI.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/BasicTextPaneUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicCaret.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicTextUI.html)    [**NO FRAMES**](http://docs.google.com/BasicTextUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

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

Class BasicTextUI

[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.TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html)  
 **javax.swing.plaf.basic.BasicTextUI**

**All Implemented Interfaces:** [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) **Direct Known Subclasses:** [BasicEditorPaneUI](http://docs.google.com/javax/swing/plaf/basic/BasicEditorPaneUI.html), [BasicTextAreaUI](http://docs.google.com/javax/swing/plaf/basic/BasicTextAreaUI.html), [BasicTextFieldUI](http://docs.google.com/javax/swing/plaf/basic/BasicTextFieldUI.html), [DefaultTextUI](http://docs.google.com/javax/swing/text/DefaultTextUI.html)

public abstract class **BasicTextUI**extends [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html)implements [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html)

Basis of a text components look-and-feel. This provides the basic editor view and controller services that may be useful when creating a look-and-feel for an extension of JTextComponent.

Most state is held in the associated JTextComponent as bound properties, and the UI installs default values for the various properties. This default will install something for all of the properties. Typically, a LAF implementation will do more however. At a minimum, a LAF would generally install key bindings.

This class also provides some concurrency support if the Document associated with the JTextComponent is a subclass of AbstractDocument. Access to the View (or View hierarchy) is serialized between any thread mutating the model and the Swing event thread (which is expected to render, do model/view coordinate translation, etc). *Any access to the root view should first acquire a read-lock on the AbstractDocument and release that lock in a finally block.*

An important method to define is the [getPropertyPrefix()](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getPropertyPrefix()) method which is used as the basis of the keys used to fetch defaults from the UIManager. The string should reflect the type of TextUI (eg. TextField, TextArea, etc) without the particular LAF part of the name (eg Metal, Motif, etc).

To build a view of the model, one of the following strategies can be employed.

1. One strategy is to simply redefine the ViewFactory interface in the UI. By default, this UI itself acts as the factory for View implementations. This is useful for simple factories. To do this reimplement the [create(javax.swing.text.Element)](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#create(javax.swing.text.Element)) method.
2. A common strategy for creating more complex types of documents is to have the EditorKit implementation return a factory. Since the EditorKit ties all of the pieces necessary to maintain a type of document, the factory is typically an important part of that and should be produced by the EditorKit implementation.

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeansTM has been added to the java.beans package. Please see [XMLEncoder](http://docs.google.com/java/beans/XMLEncoder.html).

| **Nested Class Summary** | |
| --- | --- |
| static class | [**BasicTextUI.BasicCaret**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicCaret.html) |
| static class | [**BasicTextUI.BasicHighlighter**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicHighlighter.html) |

| **Constructor Summary** | |
| --- | --- |
| [**BasicTextUI**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#BasicTextUI())()            Creates a new UI. |

| **Method Summary** | |
| --- | --- |
| [View](http://docs.google.com/javax/swing/text/View.html) | [**create**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#create(javax.swing.text.Element))([Element](http://docs.google.com/javax/swing/text/Element.html) elem)            Creates a view for an element. |
| [View](http://docs.google.com/javax/swing/text/View.html) | [**create**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#create(javax.swing.text.Element,%20int,%20int))([Element](http://docs.google.com/javax/swing/text/Element.html) elem, int p0, int p1)            Creates a view for an element. |
| protected  [Caret](http://docs.google.com/javax/swing/text/Caret.html) | [**createCaret**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#createCaret())()            Creates the object to use for a caret. |
| protected  [Highlighter](http://docs.google.com/javax/swing/text/Highlighter.html) | [**createHighlighter**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#createHighlighter())()            Creates the object to use for adding highlights. |
| protected  [Keymap](http://docs.google.com/javax/swing/text/Keymap.html) | [**createKeymap**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#createKeymap())()            Creates the keymap to use for the text component, and installs any necessary bindings into it. |
| void | [**damageRange**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#damageRange(javax.swing.text.JTextComponent,%20int,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc, int p0, int p1)            Causes the portion of the view responsible for the given part of the model to be repainted. |
| void | [**damageRange**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#damageRange(javax.swing.text.JTextComponent,%20int,%20int,%20javax.swing.text.Position.Bias,%20javax.swing.text.Position.Bias))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) t, int p0, int p1, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) p0Bias, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) p1Bias)            Causes the portion of the view responsible for the given part of the model to be repainted. |
| protected  [JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) | [**getComponent**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getComponent())()            Fetches the text component associated with this UI implementation. |
| [EditorKit](http://docs.google.com/javax/swing/text/EditorKit.html) | [**getEditorKit**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getEditorKit(javax.swing.text.JTextComponent))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc)            Fetches the EditorKit for the UI. |
| protected  [String](http://docs.google.com/java/lang/String.html) | [**getKeymapName**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getKeymapName())()            Fetches the name of the keymap that will be installed/used by default for this UI. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMaximumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getMaximumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Gets the maximum size for the editor component. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getMinimumSize**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getMinimumSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Gets the minimum size for the editor component. |
| int | [**getNextVisualPositionFrom**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getNextVisualPositionFrom(javax.swing.text.JTextComponent,%20int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias%5B%5D))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) t, int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) b, 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. |
| [Dimension](http://docs.google.com/java/awt/Dimension.html) | [**getPreferredSize**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getPreferredSize(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Gets the preferred size for the editor component. |
| protected abstract  [String](http://docs.google.com/java/lang/String.html) | [**getPropertyPrefix**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getPropertyPrefix())()            Gets the name used as a key to look up properties through the UIManager. |
| [View](http://docs.google.com/javax/swing/text/View.html) | [**getRootView**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getRootView(javax.swing.text.JTextComponent))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc)            Fetches a View with the allocation of the associated text component (i.e. |
| [String](http://docs.google.com/java/lang/String.html) | [**getToolTipText**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getToolTipText(javax.swing.text.JTextComponent,%20java.awt.Point))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) t, [Point](http://docs.google.com/java/awt/Point.html) pt)            Returns the string to be used as the tooltip at the passed in location. |
| protected  [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**getVisibleEditorRect**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getVisibleEditorRect())()            Gets the allocation to give the root View. |
| protected  void | [**installDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installDefaults())()            Initializes component properties, e.g. |
| protected  void | [**installKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installKeyboardActions())() |
| protected  void | [**installListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installListeners())()            Installs listeners for the UI. |
| void | [**installUI**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Installs the UI for a component. |
| protected  void | [**modelChanged**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#modelChanged())()            Flags model changes. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**modelToView**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#modelToView(javax.swing.text.JTextComponent,%20int))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc, int pos)            Converts the given location in the model to a place in the view coordinate system. |
| [Rectangle](http://docs.google.com/java/awt/Rectangle.html) | [**modelToView**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#modelToView(javax.swing.text.JTextComponent,%20int,%20javax.swing.text.Position.Bias))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc, int pos, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)            Converts the given location in the model to a place in the view coordinate system. |
| void | [**paint**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.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 interface. |
| protected  void | [**paintBackground**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#paintBackground(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Paints a background for the view. |
| protected  void | [**paintSafely**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#paintSafely(java.awt.Graphics))([Graphics](http://docs.google.com/java/awt/Graphics.html) g)            Paints the interface safely with a guarantee that the model won't change from the view of this thread. |
| protected  void | [**propertyChange**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#propertyChange(java.beans.PropertyChangeEvent))([PropertyChangeEvent](http://docs.google.com/java/beans/PropertyChangeEvent.html) evt)            This method gets called when a bound property is changed on the associated JTextComponent. |
| protected  void | [**setView**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#setView(javax.swing.text.View))([View](http://docs.google.com/javax/swing/text/View.html) v)            Sets the current root of the view hierarchy and calls invalidate(). |
| protected  void | [**uninstallDefaults**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallDefaults())()            Sets the component properties that haven't been explicitly overridden to null. |
| protected  void | [**uninstallKeyboardActions**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallKeyboardActions())() |
| protected  void | [**uninstallListeners**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallListeners())()            Uninstalls listeners for the UI. |
| void | [**uninstallUI**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallUI(javax.swing.JComponent))([JComponent](http://docs.google.com/javax/swing/JComponent.html) c)            Deinstalls the UI for a component. |
| void | [**update**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#update(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)            Superclass paints background in an uncontrollable way (i.e. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc, [Point](http://docs.google.com/java/awt/Point.html) pt)            Converts the given place in the view coordinate system to the nearest representative location in the model. |
| int | [**viewToModel**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point,%20javax.swing.text.Position.Bias%5B%5D))([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc, [Point](http://docs.google.com/java/awt/Point.html) pt, [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasReturn)            Converts the given place in the view coordinate system to the nearest representative location in the model. |

| **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)), [createUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html#createUI(javax.swing.JComponent)), [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)) |

| **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** |
| --- |

### BasicTextUI

public **BasicTextUI**()

Creates a new UI.

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

### createCaret

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

Creates the object to use for a caret. By default an instance of BasicCaret is created. This method can be redefined to provide something else that implements the InputPosition interface or a subclass of JCaret.

**Returns:**the caret object

### createHighlighter

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

Creates the object to use for adding highlights. By default an instance of BasicHighlighter is created. This method can be redefined to provide something else that implements the Highlighter interface or a subclass of DefaultHighlighter.

**Returns:**the highlighter

### getKeymapName

protected [String](http://docs.google.com/java/lang/String.html) **getKeymapName**()

Fetches the name of the keymap that will be installed/used by default for this UI. This is implemented to create a name based upon the classname. The name is the the name of the class with the package prefix removed.

**Returns:**the name

### createKeymap

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

Creates the keymap to use for the text component, and installs any necessary bindings into it. By default, the keymap is shared between all instances of this type of TextUI. The keymap has the name defined by the getKeymapName method. If the keymap is not found, then DEFAULT\_KEYMAP from JTextComponent is used.

The set of bindings used to create the keymap is fetched from the UIManager using a key formed by combining the [getPropertyPrefix()](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getPropertyPrefix()) method and the string .keyBindings. The type is expected to be JTextComponent.KeyBinding[].

**Returns:**the keymap**See Also:**[getKeymapName()](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#getKeymapName()), [JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html)

### propertyChange

protected void **propertyChange**([PropertyChangeEvent](http://docs.google.com/java/beans/PropertyChangeEvent.html) evt)

This method gets called when a bound property is changed on the associated JTextComponent. This is a hook which UI implementations may change to reflect how the UI displays bound properties of JTextComponent subclasses. This is implemented to do nothing (i.e. the response to properties in JTextComponent itself are handled prior to calling this method). This implementation updates the background of the text component if the editable and/or enabled state changes.

**Parameters:**evt - the property change event

### getPropertyPrefix

protected abstract [String](http://docs.google.com/java/lang/String.html) **getPropertyPrefix**()

Gets the name used as a key to look up properties through the UIManager. This is used as a prefix to all the standard text properties.

**Returns:**the name

### installDefaults

protected void **installDefaults**()

Initializes component properties, e.g. font, foreground, background, caret color, selection color, selected text color, disabled text color, and border color. The font, foreground, and background properties are only set if their current value is either null or a UIResource, other properties are set if the current value is null.

**See Also:**[uninstallDefaults()](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallDefaults()), [installUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installUI(javax.swing.JComponent))

### uninstallDefaults

protected void **uninstallDefaults**()

Sets the component properties that haven't been explicitly overridden to null. A property is considered overridden if its current value is not a UIResource.

**See Also:**[installDefaults()](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#installDefaults()), [uninstallUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.html#uninstallUI(javax.swing.JComponent))

### installListeners

protected void **installListeners**()

Installs listeners for the UI.

### uninstallListeners

protected void **uninstallListeners**()

Uninstalls listeners for the UI.

### installKeyboardActions

protected void **installKeyboardActions**()

### uninstallKeyboardActions

protected void **uninstallKeyboardActions**()

### paintBackground

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

Paints a background for the view. This will only be called if isOpaque() on the associated component is true. The default is to paint the background color of the component.

**Parameters:**g - the graphics context

### getComponent

protected final [JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) **getComponent**()

Fetches the text component associated with this UI implementation. This will be null until the ui has been installed.

**Returns:**the editor component

### modelChanged

protected void **modelChanged**()

Flags model changes. This is called whenever the model has changed. It is implemented to rebuild the view hierarchy to represent the default root element of the associated model.

### setView

protected final void **setView**([View](http://docs.google.com/javax/swing/text/View.html) v)

Sets the current root of the view hierarchy and calls invalidate(). If there were any child components, they will be removed (i.e. there are assumed to have come from components embedded in views).

**Parameters:**v - the root view

### paintSafely

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

Paints the interface safely with a guarantee that the model won't change from the view of this thread. This does the following things, rendering from back to front.

1. If the component is marked as opaque, the background is painted in the current background color of the component.
2. The highlights (if any) are painted.
3. The view hierarchy is painted.
4. The caret is painted.

**Parameters:**g - the graphics context

### installUI

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

Installs the UI for a component. This does the following things.

1. Set the associated component to opaque (can be changed easily by a subclass or on JTextComponent directly), which is the most common case. This will cause the component's background color to be painted.
2. Install the default caret and highlighter into the associated component.
3. Attach to the editor and model. If there is no model, a default one is created.
4. create the view factory and the view hierarchy used to represent the model.

**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 editor component**See Also:**[ComponentUI.installUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#installUI(javax.swing.JComponent))

### uninstallUI

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

Deinstalls the UI for a component. This removes the listeners, uninstalls the highlighter, removes views, and nulls out the keymap.

**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 editor component**See Also:**[ComponentUI.uninstallUI(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#uninstallUI(javax.swing.JComponent))

### update

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

Superclass paints background in an uncontrollable way (i.e. one might want an image tiled into the background). To prevent this from happening twice, this method is reimplemented to simply paint.

*NOTE:* Superclass is also not thread-safe in it's rendering of the background, although that's not an issue with the default rendering.

**Overrides:**[update](http://docs.google.com/javax/swing/plaf/ComponentUI.html#update(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.paint(java.awt.Graphics, javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#paint(java.awt.Graphics,%20javax.swing.JComponent)), [JComponent.paintComponent(java.awt.Graphics)](http://docs.google.com/javax/swing/JComponent.html#paintComponent(java.awt.Graphics))

### paint

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

Paints the interface. This is routed to the paintSafely method under the guarantee that the model won't change from the view of this thread while it's rendering (if the associated model is derived from AbstractDocument). This enables the model to potentially be updated asynchronously.

**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 contextc - the editor component**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)

Gets the preferred size for the editor component. If the component has been given a size prior to receiving this request, it will set the size of the view hierarchy to reflect the size of the component before requesting the preferred size of the view hierarchy. This allows formatted views to format to the current component size before answering the request. Other views don't care about currently formatted size and give the same answer either way.

**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 editor component **Returns:**the size**See Also:**[JComponent.getPreferredSize()](http://docs.google.com/javax/swing/JComponent.html#getPreferredSize()), [LayoutManager.preferredLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#preferredLayoutSize(java.awt.Container))

### getMinimumSize

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

Gets the minimum size for the editor component.

**Overrides:**[getMinimumSize](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getMinimumSize(javax.swing.JComponent)) in class [ComponentUI](http://docs.google.com/javax/swing/plaf/ComponentUI.html) **Parameters:**c - the editor component **Returns:**the size**See Also:**[JComponent.getMinimumSize()](http://docs.google.com/javax/swing/JComponent.html#getMinimumSize()), [LayoutManager.minimumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager.html#minimumLayoutSize(java.awt.Container)), [ComponentUI.getPreferredSize(javax.swing.JComponent)](http://docs.google.com/javax/swing/plaf/ComponentUI.html#getPreferredSize(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)

Gets the maximum size for the editor component.

**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 editor component **Returns:**the size**See Also:**[JComponent.getMaximumSize()](http://docs.google.com/javax/swing/JComponent.html#getMaximumSize()), [LayoutManager2.maximumLayoutSize(java.awt.Container)](http://docs.google.com/java/awt/LayoutManager2.html#maximumLayoutSize(java.awt.Container))

### getVisibleEditorRect

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

Gets the allocation to give the root View. Due to an unfortunate set of historical events this method is inappropriately named. The Rectangle returned has nothing to do with visibility. The component must have a non-zero positive size for this translation to be computed.

**Returns:**the bounding box for the root view

### modelToView

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **modelToView**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc,  
 int pos)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Converts the given location in the model to a place in the view coordinate system. The component must have a non-zero positive size for this translation to be computed.

**Specified by:**[modelToView](http://docs.google.com/javax/swing/plaf/TextUI.html#modelToView(javax.swing.text.JTextComponent,%20int)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installedpos - the local location in the model to translate >= 0 **Returns:**the coordinates as a rectangle, null if the model is not painted **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**See Also:**[TextUI.modelToView(javax.swing.text.JTextComponent, int)](http://docs.google.com/javax/swing/plaf/TextUI.html#modelToView(javax.swing.text.JTextComponent,%20int))

### modelToView

public [Rectangle](http://docs.google.com/java/awt/Rectangle.html) **modelToView**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc,  
 int pos,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) bias)  
 throws [BadLocationException](http://docs.google.com/javax/swing/text/BadLocationException.html)

Converts the given location in the model to a place in the view coordinate system. The component must have a non-zero positive size for this translation to be computed.

**Specified by:**[modelToView](http://docs.google.com/javax/swing/plaf/TextUI.html#modelToView(javax.swing.text.JTextComponent,%20int,%20javax.swing.text.Position.Bias)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installedpos - the local location in the model to translate >= 0 **Returns:**the coordinates as a rectangle, null if the model is not painted **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**See Also:**[TextUI.modelToView(javax.swing.text.JTextComponent, int)](http://docs.google.com/javax/swing/plaf/TextUI.html#modelToView(javax.swing.text.JTextComponent,%20int))

### viewToModel

public int **viewToModel**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc,  
 [Point](http://docs.google.com/java/awt/Point.html) pt)

Converts the given place in the view coordinate system to the nearest representative location in the model. The component must have a non-zero positive size for this translation to be computed.

**Specified by:**[viewToModel](http://docs.google.com/javax/swing/plaf/TextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installedpt - the location in the view to translate. This should be in the same coordinate system as the mouse events. **Returns:**the offset from the start of the document >= 0, -1 if not painted**See Also:**[TextUI.viewToModel(javax.swing.text.JTextComponent, java.awt.Point)](http://docs.google.com/javax/swing/plaf/TextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point))

### viewToModel

public int **viewToModel**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc,  
 [Point](http://docs.google.com/java/awt/Point.html) pt,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html)[] biasReturn)

Converts the given place in the view coordinate system to the nearest representative location in the model. The component must have a non-zero positive size for this translation to be computed.

**Specified by:**[viewToModel](http://docs.google.com/javax/swing/plaf/TextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point,%20javax.swing.text.Position.Bias%5B%5D)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installedpt - the location in the view to translate. This should be in the same coordinate system as the mouse events.biasReturn - filled in by this method to indicate whether the point given is closer to the previous or the next character in the model **Returns:**the offset from the start of the document >= 0, -1 if the component doesn't yet have a positive size.**See Also:**[TextUI.viewToModel(javax.swing.text.JTextComponent, java.awt.Point)](http://docs.google.com/javax/swing/plaf/TextUI.html#viewToModel(javax.swing.text.JTextComponent,%20java.awt.Point))

### getNextVisualPositionFrom

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

**Specified by:**[getNextVisualPositionFrom](http://docs.google.com/javax/swing/plaf/TextUI.html#getNextVisualPositionFrom(javax.swing.text.JTextComponent,%20int,%20javax.swing.text.Position.Bias,%20int,%20javax.swing.text.Position.Bias%5B%5D)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**t - the text component for which this UI is installedpos - the position to convert >= 0b - the bias for the positiondirection - the direction from the current position that can be thought of as the arrow keys typically found on a keyboard. This may be SwingConstants.WEST, SwingConstants.EAST, SwingConstants.NORTH, or SwingConstants.SOUTHbiasRet - an array to contain the bias for the returned position **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)

### damageRange

public void **damageRange**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc,  
 int p0,  
 int p1)

Causes the portion of the view responsible for the given part of the model to be repainted. Does nothing if the view is not currently painted.

**Specified by:**[damageRange](http://docs.google.com/javax/swing/plaf/TextUI.html#damageRange(javax.swing.text.JTextComponent,%20int,%20int)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installedp0 - the beginning of the range >= 0p1 - the end of the range >= p0**See Also:**[TextUI.damageRange(javax.swing.text.JTextComponent, int, int)](http://docs.google.com/javax/swing/plaf/TextUI.html#damageRange(javax.swing.text.JTextComponent,%20int,%20int))

### damageRange

public void **damageRange**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) t,  
 int p0,  
 int p1,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) p0Bias,  
 [Position.Bias](http://docs.google.com/javax/swing/text/Position.Bias.html) p1Bias)

Causes the portion of the view responsible for the given part of the model to be repainted.

**Specified by:**[damageRange](http://docs.google.com/javax/swing/plaf/TextUI.html#damageRange(javax.swing.text.JTextComponent,%20int,%20int,%20javax.swing.text.Position.Bias,%20javax.swing.text.Position.Bias)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**p0 - the beginning of the range >= 0p1 - the end of the range >= p0

### getEditorKit

public [EditorKit](http://docs.google.com/javax/swing/text/EditorKit.html) **getEditorKit**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc)

Fetches the EditorKit for the UI.

**Specified by:**[getEditorKit](http://docs.google.com/javax/swing/plaf/TextUI.html#getEditorKit(javax.swing.text.JTextComponent)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installed **Returns:**the editor capabilities**See Also:**[TextUI.getEditorKit(javax.swing.text.JTextComponent)](http://docs.google.com/javax/swing/plaf/TextUI.html#getEditorKit(javax.swing.text.JTextComponent))

### getRootView

public [View](http://docs.google.com/javax/swing/text/View.html) **getRootView**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) tc)

Fetches a View with the allocation of the associated text component (i.e. the root of the hierarchy) that can be traversed to determine how the model is being represented spatially.

**NOTE:**The View hierarchy can be traversed from the root view, and other things can be done as well. Things done in this way cannot be protected like simple method calls through the TextUI. Therefore, proper operation in the presence of concurrency must be arranged by any logic that calls this method!

**Specified by:**[getRootView](http://docs.google.com/javax/swing/plaf/TextUI.html#getRootView(javax.swing.text.JTextComponent)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Parameters:**tc - the text component for which this UI is installed **Returns:**the view**See Also:**[TextUI.getRootView(javax.swing.text.JTextComponent)](http://docs.google.com/javax/swing/plaf/TextUI.html#getRootView(javax.swing.text.JTextComponent))

### getToolTipText

public [String](http://docs.google.com/java/lang/String.html) **getToolTipText**([JTextComponent](http://docs.google.com/javax/swing/text/JTextComponent.html) t,  
 [Point](http://docs.google.com/java/awt/Point.html) pt)

Returns the string to be used as the tooltip at the passed in location. This forwards the method onto the root View.

**Overrides:**[getToolTipText](http://docs.google.com/javax/swing/plaf/TextUI.html#getToolTipText(javax.swing.text.JTextComponent,%20java.awt.Point)) in class [TextUI](http://docs.google.com/javax/swing/plaf/TextUI.html) **Since:** 1.4 **See Also:**[JTextComponent.getToolTipText(java.awt.event.MouseEvent)](http://docs.google.com/javax/swing/text/JTextComponent.html#getToolTipText(java.awt.event.MouseEvent)), [View.getToolTipText(float, float, java.awt.Shape)](http://docs.google.com/javax/swing/text/View.html#getToolTipText(float,%20float,%20java.awt.Shape))

### create

public [View](http://docs.google.com/javax/swing/text/View.html) **create**([Element](http://docs.google.com/javax/swing/text/Element.html) elem)

Creates a view for an element. If a subclass wishes to directly implement the factory producing the view(s), it should reimplement this method. By default it simply returns null indicating it is unable to represent the element.

**Specified by:**[create](http://docs.google.com/javax/swing/text/ViewFactory.html#create(javax.swing.text.Element)) in interface [ViewFactory](http://docs.google.com/javax/swing/text/ViewFactory.html) **Parameters:**elem - the element **Returns:**the view**See Also:**[View](http://docs.google.com/javax/swing/text/View.html)

### create

public [View](http://docs.google.com/javax/swing/text/View.html) **create**([Element](http://docs.google.com/javax/swing/text/Element.html) elem,  
 int p0,  
 int p1)

Creates a view for an element. If a subclass wishes to directly implement the factory producing the view(s), it should reimplement this method. By default it simply returns null indicating it is unable to represent the part of the element.

**Parameters:**elem - the elementp0 - the starting offset >= 0p1 - the ending offset >= p0 **Returns:**the view

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BasicTextUI.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/BasicTextPaneUI.html)   [**NEXT CLASS**](http://docs.google.com/javax/swing/plaf/basic/BasicTextUI.BasicCaret.html) | [**FRAMES**](http://docs.google.com/index.html?javax/swing/plaf/basic/BasicTextUI.html)    [**NO FRAMES**](http://docs.google.com/BasicTextUI.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: [NESTED](#3znysh7) | FIELD | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: FIELD | [CONSTR](#4d34og8) | [METHOD](#17dp8vu) |

[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).