NSTextField

Inherits From:	NSControl : NSView : NSResponder : NSObject
Conforms To:	NSCoding (NSResponder) NSObject (NSObject)
Declared In:	AppKit/NSTextField.h

Class Description

An NSTextField is a kind of NSControl that displays text that the user can select or edit, and which sends its action message to its target when the user presses the Return key while editing. Like other controls, it also performs validation on its value when edited; if the value isn't valid it sends a special error action message to its target. An NSTextField can be assigned a delegate, who is then informed of delegate messages sent by the window's field editor, such as **textShouldEndEditing:**. See the NSWindow and NSTextView class specifications for more information on a window's field editor.

Typical of several kinds of control, NSTextField allows you to set its text and background color, whether it draws the background, and whether it draws a bezel or border around its text. You can also link text fields together in their window's key view loop, as described in the NSWindow class specification.

Method Types

Controlling editability and selectability	– setEditable: – isEditable – setSelectable: – isSelectable
Setting the error action	– setErrorAction: – errorAction
Controlling rich text behavior	– setAllowsEditingTextAttriutes: – allowsEditingTextAttributes – setImportsGraphics: – importsGraphics
Setting the text color	– setTextColor: – textColor

Controlling the background	– setBackgroundColor: – backgroundColor – setDrawsBackground: – drawsBackground
Setting a border	– setBezeled: – isBezeled – setBordered: – isBordered
Linking text fields together	– setNextText: – nextText – setPreviousText: – previousText
Selecting the text	– selectText:
Working with the responder chain	– acceptsFirstResponder
Using keyboard interface control	– setTitleWithMnemonic: – performMnemonic:
Setting the delegate	– setDelegate: – delegate
Text delegate methods	– textShouldBeginEditing: – textDidBeginEditing: – textDidChange: – textShouldEndEditing: – textDidEndEditing:

Instance Methods

acceptsFirstResponder

– (BOOL)**acceptsFirstResponder**

Returns YES if the receiver is editable or selectable, NO otherwise.

allowsEditingTextAttributes

– (BOOL)**allowsEditingTextAttributes**

Returns YES if the receiver allows the user to change font attributes of the receiver's text, NO if the user isn't permitted to do so. You can change text attributes programmatically regardless of this setting.

See also: – **importsGraphics**, – **setAllowsEditingTextAttributes:**

backgroundColor

– (NSColor *)**backgroundColor**

Returns the color of the background that the receiver draws behind the text.

See also: – **drawsBackground**, – **backgroundColor** (NSCell), – **setBackgroundColor:**

delegate

– (id)**delegate**

Returns the receiver's delegate.

See also: – **textShouldBeginEditing:**, – **textShouldEndEditing:**, – **textDidBeginEditing:**, – **textDidEndEditing:**, – **textDidChange:**, – **setDelegate:**

drawsBackground

– (BOOL)**drawsBackground**

Returns YES if the receiver's cell draws its background color behind its text, NO if it draws no background.

See also: – **backgroundColor**, – **drawsBackground** (NSTextFieldCell), – **setDrawsBackground:**

errorAction

– (SEL)**errorAction**

Returns the selector for the message sent to the receiver's target whenever a validation error occurs.

See also: – **textShouldEndEditing:**, – **setErrorAction:**

importsGraphics

– (BOOL)**importsGraphics**

Returns YES if the receiver allows the user to drag image files into it, NO if it doesn't accept dragged images. You can add images programmatically regardless of this setting.

See also: – **allowsEditingTextAttributes**, – **importsGraphics** (NSTextView), – **setImportsGraphics:**

isBezeled

– (BOOL)**isBezeled**

Returns YES if the receiver draws a bezeled frame around its contents, NO if it doesn't.

See also: – **isBordered**, – **setBezeled:**

isBordered

– (BOOL)**isBordered**

Returns YES if the receiver draws a solid black border around its contents, NO if it doesn't.

See also: – **isBezeled**, – **setBordered:**

isEditable

– (BOOL)**isEditable**

Returns YES if the user is allowed to select and edit the receiver's text, NO if the user isn't allowed to edit it (though the user may be able to select it).

See also: – **isSelectable**, – **setEditable:**

isSelectable

– (BOOL)**isSelectable**

Returns YES if the user is allowed to select the receiver's text, NO if the user isn't allowed to select it. Selectable text isn't necessarily editable; use **isEditable** to check for editability.

See also: – **setSelectable:**

nextText

– (id)**nextText**

Returns the receiver's next key view, the object that's made first responder when the user presses the Tab key while editing the receiver. See the description of the **nextKeyView** method in NSView for more information.

See also: – **previousText**, – **setNextText:**

performMnemonic:

– (BOOL)**performMnemonic:**(NSString *)*aString*

Does nothing and returns NO if the receiver is editable. Otherwise makes the receiver's next key view the first responder and returns YES. This allows a text field to be used as a label for another interface object, so that it moves control to that object when its mnemonic is invoked.

See also: – **setNextKeyView:** (NSView), – **isEditable**, – **setTitleWithMnemonic:**, – **makeFirstResponder:** (NSWindow)

previousText

– (id)**previousText**

Returns the receiver's previous key view, the object that's made first responder when the user presses Shift-Tab while editing the receiver. See the description of the **previousKeyView** method in NSView for more information.

See also: – **nextText**, – **setPreviousText:**

selectText:

– (void)**selectText:**(id)*sender*

Selects the entire contents of the receiver if it's selectable. However, if the receiver isn't in some window's view hierarchy, this method has no effect.

See also: – **isSelectable**

setAllowsEditingTextAttributes:

– (void)**setAllowsEditingTextAttributes:**(BOOL)*flag*

Controls whether the receiver allows the user to change font attributes of the receiver's text. If *flag* is YES, the user is permitted to make such changes; if *flag* is NO, the user isn't so permitted. You can change text attributes programmatically regardless of this setting.

See also: – **setImportsGraphics:**, – **allowsEditingTextAttributes**

setBackgroundColor:

– (void)**setBackgroundColor:**(NSColor *)*aColor*

Sets the color of the background that the receiver draws behind the text to *aColor*.

See also: – **setDrawsBackground:**, – **setBackgroundColor:** (NSCell) – **backgroundColor**

setBezeled:

– (void)**setBezeled:**(BOOL)*flag*

Controls whether the receiver draws a bezeled border around its contents. If *flag* is YES, then it draws a bezeled border; if *flag* is NO, it draws no border.

See also: – **isBezeled**, – **setBordered:**

setBordered:

– (void)**setBordered:**(BOOL)*flag*

Controls whether the receiver draws a solid black border around its contents. If *flag* is YES, then it draws a border; if *flag* is NO, it draws no border.

See also: – **isBordered**, – **setBezeled:**

setDelegate:

– (void)**setDelegate:**(id)*anObject*

Sets the receiver's delegate to *anObject*.

See also: – **textShouldBeginEditing:**, – **textShouldEndEditing:**, – **textDidBeginEditing:**, – **textDidEndEditing:**, – **textDidChange:**, – **delegate**

setDrawsBackground:

– (void)**setDrawsBackground:**(BOOL)*flag*

Controls whether the receiver draws its background color behind its text. If *flag* is YES, then it does; if *flag* is NO, then it draws nothing behind its text.

See also: – **setBackgroundColor:**, – **setDrawsBackground:** (NSTextFieldCell), – **drawsBackground**

setEditable:

– (void)**setEditable:**(BOOL)*flag*

Controls whether the user can edits the receiver's text. If *flag* is YES, then the user is allowed to both select and edit text. If *flag* is NO, then the user isn't permitted to edit text, and the receiver's selectability is restored to its previous value. For example, if an NSTextField is selectable but not editable, then made editable for a time, then made not editable, it remains selectable. To guarantee that text is neither editable nor selectable, simply use **setSelectable:** to turn off selectability.

See also: – **isEditable**

setErrorAction:

– (void)**setErrorAction:**(SEL)*aSelector*

Sets the selector for the message sent to the receiver's target whenever a validation error occurs to *aSelector*.

See also: – **textShouldEndEditing:**, – **errorAction**

setImportsGraphics:

– (void)**setImportsGraphics:**(BOOL)*flag*

Controls whether the receiver allows the user to drag image files into it. If *flag* is YES, the receiver accepts dragged images; if *flag* is NO, it doesn't. You can add images programmatically regardless of this setting.

See also: – **setAllowsEditingTextAttributes:**, – **setImportsGraphics** (NSTextView), – **importsGraphics**

setNextText:

– (void)**setNextText:**(id)*anObject*

Sets the receiver's next key view to *anObject*, which should be a kind of NSView. See the description of the **setNextKeyView:** method in the NSView class specification for more information.

See also: – **setPreviousText:**, – **nextText**

setPreviousText:

– (void)**setPreviousText:**(id)*anObject*

Sets the receiver's previous key view to *anObject*, which should be a kind of NSView. See the description of the **setPreviousKeyView:** method in the NSView class specification for more information.

See also: – **setNextText:**, – **previousText**

setSelectable:

– (void)**setSelectable:**(BOOL)*flag*

Returns YES if the user is allowed to select the receiver's text, NO if the user isn't allowed to select it. Selectable text isn't necessarily editable; use **isEditable** to check for editability.

See also: – **setSelectable:**

setTextColor:

– (void)**setTextColor:**(NSColor *)*aColor*

Sets the color used to draw the receiver's text to *aColor*.

See also: – **setBackgroundColor:**, **– setTextColor:** (NSTextFieldCell), **– textColor**

setTitleWithMnemonic:

– (void)**setTitleWithMnemonic:**(NSString *)*aString*

Sets the receiver's string value to *aString*, using the first character preceded by an ampersand ('&') as the mnemonic and stripping out that first ampersand character. Use this method only with a non-editable text field being used as a label for another interface component, which you should establish using **setNextKeyView:**. When set up in this fashion, the text field's mnemonic serves to select the other interface component.

See also: – **performMnemonic:**

textColor

– (NSColor *)**textColor**

Returns the color used to draw the receiver's text.

See also: – **backgroundColor**, – **textColor** (NSTextFieldCell), – **setTextColor:**

textDidBeginEditing:

– (void)**textDidBeginEditing:**(NSNotification *)*aNotification*

Posts an NSControlTextDidBeginEditingNotification to the default notification center. This causes the receiver's delegate to receive a **control:textDidBeginEditing:** message. See the NSControl class specification for more information on the text delegate method.

See also: – **textDidBeginEditing:**, – **textDidChange:**, – **textShouldEndEditing:**, – **textDidEndEditing:**

textDidChange:

– (void)**textDidChange:**(NSNotification *)*aNotification*

Forwards this message to the receiver's cell if it responds, and posts an NSControlTextDidChangeNotification to the default notification center. This causes the receiver's delegate to receive a **control:textDidChange:** message. See the NSControl class specification for more information on the text delegate method.

See also: – **textShouldBeginEditing:**, – **textDidBeginEditing:**, – **textShouldEndEditing:**, – **textDidEndEditing:**

textDidEndEditing:

– (void)**textDidEndEditing:**(NSNotification *)*aNotification*

Handles an end to editing. After validating the new value, posts an NSControlTextDidEndEditingNotification to the default notification center. This causes the receiver's delegate to receive a **control:textDidEndEditing:** message. After this, sends **endEditing:** to the receiver' cell, and handles the key that caused editing to end:

- If the user ended editing by pressing Return, this method tries to send the receiver's action to its target; if unsuccessful, it sends **performKeyEquivalent:** to its NSWindow (for example, to handle the default button on a panel); if that also fails, then the receiver simply selects its text.
- If the user ended editing by pressing Tab or Shift-Tab, the receiver tries to have its NSWindow select its next or previous key view, using the NSWindow method **selectKeyViewFollowingView:** or **selectKeyViewPreceedingView:**. If unsuccessful in doing this, the receiver simply selects its text.

See the NSControl class specification for more information on the text delegate method.

See also: – **textShouldBeginEditing:**, – **textDidBeginEditing:**, – **textDidChange:**, – **textShouldEndEditing:**

textShouldBeginEditing:

– (BOOL)**textShouldBeginEditing:**(NSText *)*textObject*

If the receiver isn't editable, returns NO immediately. If it is editable and its delegate responds to **control:textShouldBeginEditing:**, invokes that method and returns the result. Otherwise simply returns YES to allow editing to occur. See the NSControl class specification for more information on the text delegate method.

See also: – **textDidBeginEditing:**, – **textDidChange:**, – **textShouldEndEditing:**, – **textDidEndEditing:**

textShouldEndEditing:

– (BOOL)**textShouldEndEditing:**(NSText *)*textObject*

Performs validation on the receiver's new value using NSCell's **isEntryAcceptable:**, sending the receiver's error action to its target if validation fails. If the new value is valid and the delegate responds to **control:textShouldEndEditing:**, invokes that method and returns the result, in addition beeping if the delegate returns NO. See the NSControl class specification for more information on the text delegate method.

See also: – **textShouldBeginEditing:**, – **textDidBeginEditing:**, – **textDidChange:**, – **textDidEndEditing:**, – **errorAction**