NSActionCell Class Reference

Cocoa > User Experience

2007-02-28

Contents

NSActionCell Class Reference

Inherits from	NSCell : NSObject
Conforms to	NSCoding (NSCell) NSCopying (NSCell) NSObject (NSObject)
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Action Messages
Declared in	NSActionCell.h
Related sample code	Clock Control TrackBall

Overview

An NSActionCell defines an active area inside a control (an instance of NSControl or one of its subclasses).

As an NSControl's active area, an NSActionCell does three things: it usually performs display of text or an icon; it provides the NSControl with a target and an action; and it handles mouse (cursor) tracking by properly highlighting its area and sending action messages to its target based on cursor movement.

Tasks

Configuring an NSActionCell Object

`– setAlignment:` (page 9)
  Sets the alignment of text in the receiver.

`– setBezeled:` (page 10)
  Sets whether the receiver draws itself with a bezeled border.

`– setBordered:` (page 10)
  Sets whether the receiver draws itself outlined with a plain border.

`– setEnabled:` (page 11)
  Sets whether the receiver is enabled or disabled.

- `setFloatingPointFormat:left:right:` (page 11)
  Sets the receiver’s floating-point format.
- `setFont:` (page 12)
  Sets the font to be used when the receiver displays text.
- `setImage:` (page 13)
  Sets the image to be displayed in the receiver.

Obtaining and Setting Cell Values

- `doubleValue` (page 8)
  Returns the receiver’s value as a `double` after validating any editing of cell content.
- `floatValue` (page 8)
  Returns the receiver’s value as a `float` after validating any editing of cell content.
- `intValue` (page 9)
  Returns the receiver’s value as an `int` after validating any editing of cell content.
- `integerValue` (page 8)
  Returns the receiver’s value as a 64-bit compatible integer after validating any editing of cell content.
- `stringValue` (page 14)
  Returns the receiver’s value as a string object as converted by the cell’s formatter, if one exists.
- `setObjectValue:` (page 13)
  Discards any editing of the receiver’s text and sets its object value to *`object`*.

Managing the Cell’s View

- `controlView` (page 7)
  Returns the view in which the receiver was last drawn.
- `setControlView:` (page 11)
  Sets the receiver's control view, the view in which it is drawn.

Assigning the Target and Action

- `setAction:` (page 9)
  Sets the selector used for action messages sent by the receiver's control.
- `action` (page 7)
  Returns the receiver’s action-message selector.
- `setTarget:` (page 14)
  Sets the receiver’s target object.
- `target` (page 15)
  Returns the receiver’s target object.

Assigning a Tag

`- setTag:` (page 13)
    Sets the receiver's tag.

`- tag` (page 15)
    Returns the receiver's tag.

Instance Methods

action

Returns the receiver's action-message selector.

```
- (SEL)action
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAction:` (page 9)
`- setTarget:` (page 14)
`- target` (page 15)

Related Sample Code
Clock Control

Declared In
`NSActionCell.h`

controlView

Returns the view in which the receiver was last drawn.

```
- (NSView *)controlView
```

Return Value
The returned view is normally an `NSControl` object. The method returns `nil` if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

Availability
Available in Mac OS X v10.0 and later.

Related Sample Code
AnimatedSlider

Declared In
`NSActionCell.h`

doubleValue

Returns the receiver's value as a `double` after validating any editing of cell content.

```
- (double)doubleValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– validateEditing` (NSControl)

Declared In
`NSActionCell.h`

floatValue

Returns the receiver's value as a `float` after validating any editing of cell content.

```
- (float)floatValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`– validateEditing` (NSControl)

Declared In
`NSActionCell.h`

integerValue

Returns the receiver's value as a 64-bit compatible integer after validating any editing of cell content.

```
- (NSInteger)integerValue
```

Return Value
A 64-bit compatible integer value, as defined by the `NSInteger` type.

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`NSActionCell.h`

intValue

Returns the receiver’s value as an `int` after validating any editing of cell content.

```
- (int)intValue
```

Discussion
If the receiver is not a text-type cell or the cell value is not scannable, the method returns 0.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- validateEditing` (NSControl)

Declared In
`NSActionCell.h`

setAction:

Sets the selector used for action messages sent by the receiver's control.

```
- (void)setAction:(SEL)aSelector
```

Parameters

aSelector
The selector that identifies the action method to invoke.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- action` (page 7)
`- setTarget:` (page 14)
`- target` (page 15)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSActionCell.h`

setAlignment:

Sets the alignment of text in the receiver.

```
- (void)setAlignment:(NSTextAlignment)mode
```

Parameters

`mode`

One of five constants that specifies alignment within the cell: `NSLeftTextAlignment`, `NSRightTextAlignment`, `NSCenterTextAlignment`, `NSJustifiedTextAlignment`, and `NSNaturalTextAlignment` (the default alignment for the text).

Discussion

The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSActionCell.h`

setBezeled:

Sets whether the receiver draws itself with a bezeled border.

```
- (void)setBezeled:(BOOL)flag
```

Parameters

`flag`

`YES` if the cell is to be drawn with a bezeled border, `NO` otherwise.

Discussion

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` and `setBordered:` (page 10) methods are mutually exclusive—that is, a border can be only plain or bezeled.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTKitMovieShuffler

Declared In

`NSActionCell.h`

setBordered:

Sets whether the receiver draws itself outlined with a plain border.

```
- (void)setBordered:(BOOL)flag
```

Parameters

`flag`

`YES` if the cell is to be drawn with a plain border, `NO` otherwise.

Discussion

After setting the attribute the method marks the receiver as needing redisplay. The `setBezeled:` (page 10) and `setBordered:` methods are mutually exclusive—that is, a border can be only plain or bezeled.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSActionCell.h`

setControlView:

Sets the receiver's control view, the view in which it is drawn.

```
- (void)setControlView:(NSView *)view
```

Parameters

`view`
: The view object, which is normally an NSControl *`view`*. Pass in `nil` if the receiver has no control view (usually because it hasn't yet been placed in the view hierarchy).

Discussion
The control view is typically set in the receiver's implementation of `drawWithFrame:inView:` (`NSCell`).

Availability
Available in Mac OS X v10.4 and later.

Declared In
`NSActionCell.h`

setEnabled:

Sets whether the receiver is enabled or disabled.

```
- (void)setEnabled:(BOOL)flag
```

Parameters

`flag`
: `YES` if the cell is to be enabled, `NO` otherwise

Discussion
The text of disabled cells is changed to gray. If a cell is disabled, it cannot be highlighted, cannot be edited, and does not support mouse tracking (and thus cannot participate in target-action behavior). The method marks the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`NSActionCell.h`

setFloatingPointFormat:left:right:

Sets the receiver's floating-point format.

```
- (void)setFloatingPointFormat:(BOOL)autoRange left:(NSUInteger)leftDigits
    right:(NSUInteger)rightDigits
```

Parameters

autoRange

NO if you want the receiver to places digits to the right and left of the decimal point as specified (in leftDigits and rightDigits; YES if you want it to place the digits flexibly.

leftDigits

The maximum number of digits to the left of the decimal point. The receiver might interpret this value flexibly if autoRange is YES.

rightDigits

The maximum number of digits to the right of the decimal point. The receiver might interpret this value flexibly if autoRange is YES.

Discussion

The implementation of this method is based on the NSCell method setFloatingPointFormat:left:right:. See the description of that method for details.

The NSActionCell implementation of the method supplements the NSCell implementation by marking the receiver as needing redisplay after discarding any editing changes that were being made to cell text.

Note: This method is being deprecated in favor of NSFormatter objects. For more information, see NSFormatter. This documentation is provided only for developers who need to modify older applications.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSActionCell.h

setFont:

Sets the font to be used when the receiver displays text.

```
- (void)setFont:(NSFont *)fontObj
```

Parameters

fontObj

The font object encapsulating information about the new font. If *fontObj* is nil and the receiver is a text-type cell, the font object currently held by the receiver is autoreleased.

Discussion

If the receiver is not a text-type cell, the method converts it to that type. NSActionCell supplements the NSCell implementation of this method by marking the updated cell as needing redisplay. If the receiver was converted to a text-type cell and is selected, it also updates the field editor with *fontObj*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSActionCell.h

setImage:

Sets the image to be displayed in the receiver.

```
- (void)setImage:(NSImage *)image
```

Parameters

`image`

The image for the receiver to display. If *`image`* is `nil`, the image currently displayed by the receiver is removed.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

FunkyOverlayWindow

Declared In

`NSActionCell.h`

setObjectValue:

Discards any editing of the receiver's text and sets its object value to *`object`*.

```
- (void)setObjectValue:(id < NSCopying >)object
```

Parameters

`object`

The object value to assign to the receiver.

Discussion

If the object value is afterward different from what it was before the method was invoked, the method marks the receiver as needing redisplay.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSActionCell.h`

setTag:

Sets the receiver's tag.

```
- (void)setTag:(NSInteger)anInt
```

Parameters

`anInt`

An integer tag to be associated with the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- tag` (page 15)

Declared In
`NSActionCell.h`

setTarget:

Sets the receiver’s target object.

```
- (void)setTarget:(id)anObject
```

Parameters

anObject
The object that is the target of action messages sent by the receiver's control.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- action` (page 7)
`- setAction:` (page 9)
`- target` (page 15)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSActionCell.h`

stringValue

Returns the receiver’s value as a string object as converted by the cell’s formatter, if one exists.

```
- (NSString *)stringValue
```

Discussion
If no formatter exists and the value is an `NSString`, returns the value as a plain, attributed, or localized formatted string. If the value is not an `NSString` or cannot be converted to one, returns an empty string. The method supplements the `NSCell` implementation by validating and retaining any editing changes being made to cell text.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- validateEditing` (NSControl)

Declared In
`NSActionCell.h`

tag

Returns the receiver's tag.

```
- (NSInteger)tag
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setTag:` (page 13)

Declared In
`NSActionCell.h`

target

Returns the receiver's target object.

```
- (id)target
```

Availability
Available in Mac OS X v10.0 and later.

See Also
`- action` (page 7)
`- setAction:` (page 9)
`- setTarget:` (page 14)

Related Sample Code
Clock Control

Declared In
`NSActionCell.h`

Document Revision History

This table describes the changes to *NSActionCell Class Reference*.

Date	Notes
2007-02-28	Updated for Mac OS X version 10.5.
2006-05-23	First publication of this content as a separate document.

Index