NSTextAttachmentCell Protocol Reference

Cocoa > Objective-C Language

2006-05-23

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Simultaneously published in the United States and Canada.

Contents

NSTextAttachmentCell Protocol Reference

Adopted by	NSTextAttachmentCell
Conforms to	NSObject
Framework	/System/Library/Frameworks/AppKit.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSTextAttachment.h
Companion guides	Text System Overview Text Attachment Programming Topics for Cocoa

Overview

The NSTextAttachmentCell protocol declares the interface for objects that draw text attachment icons and handle mouse events on their icons. With the exceptions of `cellBaselineOffset` (page 7), `setAttachment:` (page 9), and `attachment` (page 6), all of these methods are implemented by the NSCell class and described in that class specification.

See the NSAttributedString and NSTextView class specifications for general information on text attachments.

Tasks

Drawing

- `drawWithFrame:inView:` (page 8)
  Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused.
- `drawWithFrame:inView:characterIndex:` (page 8)
  Draws the receiver's image within *cellFrame* in *aView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text.
- `drawWithFrame:inView:characterIndex:layoutManager:` (page 8)
  Draws the receiver's image within *cellFrame* in *controlView*, which is the view currently focused. *charIndex* is the index of the attachment character within the text. *layoutManager* is the layout manager for the text.

- highlight:withFrame:inView: (page 9)
  Draws the receiver's image—with highlighting if *flag* is YES—within *cellFrame* in *aView*, which should be the focus view.

Cell Size and Position

- cellSize (page 8)
  Returns the size of the attachment's icon.
- cellBaselineOffset (page 7)
  Returns the position where the attachment cell's image should be drawn in text, relative to the current point established in the glyph layout.
- cellFrameForTextContainer:proposedLineFragment:glyphPosition:characterIndex: (page 7)
  Returns the frame of the cell as it would be drawn as the character at the given glyph *position*, and character index, *charIndex*, in *textContainer*.

Event Handling

- wantsToTrackMouse (page 11)
  Returns YES if the receiver will handle a mouse event occurring over its image (to support dragging, for example), NO otherwise.
- wantsToTrackMouseForEvent:inRect:ofView:atCharacterIndex: (page 11)
  Allows an attachment to specify what events it would want to track the mouse for.
- trackMouse:inRect:ofView:untilMouseUp: (page 10)
  Handles a mouse-down event on the receiver's image.
- trackMouse:inRect:ofView:atCharacterIndex:untilMouseUp: (page 9)
  Handles a mouse-down event on the receiver's image.

Setting the Attachment

- setAttachment: (page 9)
  Sets the text attachment object that owns the receiver to *anAttachment*, without retaining it (the text attachment, as the owner, retains the cell).
- attachment (page 6)
  Returns the text attachment object that owns the receiver.

Instance Methods

attachment

Returns the text attachment object that owns the receiver.

```
- (NSTextAttachment *)attachment
```

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

See Also
- `setAttachment:` (page 9)

Related Sample Code
Quartz Composer WWDC 2005 TextEdit
TextEditPlus

Declared In
`NSTextAttachment.h`

cellBaselineOffset

Returns the position where the attachment cell's image should be drawn in text, relative to the current point established in the glyph layout.

```
- (NSPoint)cellBaselineOffset
```

Discussion
The image should be drawn so its lower-left corner lies on this point.

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

See Also
- `icon` (NSFileWrapper)

Declared In
`NSTextAttachment.h`

cellFrameForTextContainer:proposedLineFragment:glyphPosition:characterIndex:

Returns the frame of the cell as it would be drawn as the character at the given glyph *position*, and character index, *charIndex*, in *textContainer*.

```
- (NSRect)cellFrameForTextContainer:(NSTextContainer *)textContainer
    proposedLineFragment:(NSRect)lineFrag glyphPosition:(NSPoint)position
    characterIndex:(NSUInteger)charIndex
```

Discussion
The proposed line fragment is specified by *lineFrag*.

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

Declared In
`NSTextAttachment.h`

cellSize

Returns the size of the attachment's icon.

```
- (NSSize)cellSize
```

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

See Also
- `icon` (NSFileWrapper)
- `fileWrapper` (NSTextAttachment)

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:

Draws the receiver's image within *`cellFrame`* in *`aView`*, which is the view currently focused.

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)aView
```

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

See Also
- `drawWithFrame:inView:` (NSCell)
- `lockFocus` (NSView)

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:characterIndex:

Draws the receiver's image within *`cellFrame`* in *`aView`*, which is the view currently focused. *`charIndex`* is the index of the attachment character within the text.

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)aView
    characterIndex:(NSUInteger)charIndex
```

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

Declared In
`NSTextAttachment.h`

drawWithFrame:inView:characterIndex:layoutManager:

Draws the receiver's image within *`cellFrame`* in *`controlView`*, which is the view currently focused. *`charIndex`* is the index of the attachment character within the text. *`layoutManager`* is the layout manager for the text.

```
- (void)drawWithFrame:(NSRect)cellFrame inView:(NSView *)controlView
    characterIndex:(NSUInteger)charIndex layoutManager:(NSLayoutManager
    *)layoutManager
```

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

Declared In
NSTextAttachment.h

highlight:withFrame:inView:

Draws the receiver's image—with highlighting if *flag* is YES—within *cellFrame* in *aView*, which should be the focus view.

```
- (void)highlight:(BOOL)flag withFrame:(NSRect)cellFrame inView:(NSView *)aView
```

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

See Also
- highlight:withFrame:inView: (NSCell)
- lockFocus (NSView)

Declared In
NSTextAttachment.h

setAttachment:

Sets the text attachment object that owns the receiver to *anAttachment*, without retaining it (the text attachment, as the owner, retains the cell).

```
- (void)setAttachment:(NSTextAttachment *)anAttachment
```

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

See Also
- attachment (page 6)
- setAttachmentCell: (NSTextAttachment)

Declared In
NSTextAttachment.h

trackMouse:inRect:ofView:atCharacterIndex:untilMouseUp:

Handles a mouse-down event on the receiver's image.

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)aTextView atCharacterIndex:(NSUInteger)charIndex untilMouseUp:(BOOL)flag
```

Discussion
theEvent is the mouse-down event. *cellFrame* is the region of *aTextView* in which you should track further mouse events. *charIndex* is the position in the text at which this attachment appears. *aTextView* is the view that received the event. It's assumed to be an NSTextView, and should be the focus view. If *flag* is YES, the receiver tracks the mouse until a mouse-up event occurs; if *flag* is NO, it stops tracking when a mouse-dragged event occurs outside of *cellFrame*. Returns YES if the receiver successfully finished tracking the mouse (typically through a mouse-up event), NO otherwise (such as when the mouse is dragged outside *cellFrame*).

NSTextAttachmentCell's implementation of this method calls upon *aTextView*'s delegate to handle the event. If *theEvent* is a mouse-up event for a double click, the text attachment cell sends the delegate a `textView:doubleClickedOnCell:inRect:` message and returns YES. Otherwise, depending on whether the user clicks or drags the cell, it sends the delegate a `textView:clickedOnCell:inRect:` or a `textView:draggedCell:inRect:event:` message and returns YES. NSTextAttachmentCell's implementation returns NO only if *flag* is NO and the mouse is dragged outside of *cellFrame*. The delegate methods are invoked only if the delegate responds.

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

Declared In
NSTextAttachment.h

trackMouse:inRect:ofView:untilMouseUp:

Handles a mouse-down event on the receiver's image.

```
- (BOOL)trackMouse:(NSEvent *)theEvent inRect:(NSRect)cellFrame ofView:(NSView
    *)aTextView untilMouseUp:(BOOL)flag
```

Discussion
theEvent is the mouse-down event. *cellFrame* is the region of *aTextView* in which further mouse events should be tracked. *aTextView* is the view that received the event. It's assumed to be an NSTextView and should be the focus view. If *flag* is YES, the receiver tracks the mouse until a mouse-up event occurs; if *flag* is NO, it stops tracking when a mouse-dragged event occurs outside of *cellFrame*. Returns YES if the receiver successfully finished tracking the mouse (typically through a mouse-up event), NO otherwise (such as when the cursor is dragged outside *cellFrame*).

NSTextAttachmentCell's implementation of this method calls upon the delegate of *aTextView* to handle the event. If *theEvent* is a mouse-up event for a double click, the text attachment cell sends the delegate a `textView:doubleClickedOnCell:inRect:` message and returns YES. Otherwise, depending on whether the user clicks or drags the cell, it sends the delegate a `textView:clickedOnCell:inRect:` or a `textView:draggedCell:inRect:event:` message and returns YES. NSTextAttachmentCell's implementation returns NO only if *flag* is NO and the cursor is dragged outside of *cellFrame*. The delegate methods are invoked only if the delegate responds.

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

See Also
- `wantsToTrackMouse` (page 11)
- `trackMouse:inRect:ofView:untilMouseUp:` (NSCell)
- `lockFocus` (NSView)

Declared In
NSTextAttachment.h

wantsToTrackMouse

Returns YES if the receiver will handle a mouse event occurring over its image (to support dragging, for example), NO otherwise.

```
- (BOOL)wantsToTrackMouse
```

Discussion
NSTextAttachmentCell's implementation of this method returns YES. The NSView containing the cell should invoke this method before sending a trackMouse:inRect:ofView:untilMouseUp: (page 10) message.

For an attachment in an attributed string, if the attachment cell returns NO its attachment character should be selected rather than the cell being asked to track the mouse. This results in the attachment icon behaving as any regular glyph in text.

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

Declared In
NSTextAttachment.h

wantsToTrackMouseForEvent:inRect:ofView:atCharacterIndex:

Allows an attachment to specify what events it would want to track the mouse for.

```
- (BOOL)wantsToTrackMouseForEvent:(NSEvent *)theEvent inRect:(NSRect)cellFrame
    ofView:(NSView *)controlView atCharacterIndex:(NSUInteger)charIndex
```

Discussion
theEvent is the event in question that occurred in *cellFrame* inside *controlView*. *charIndex* is the index of the attachment character within the text. If wantsToTrackMouse (page 11) returns YES, this method allows the attachment to decide whether it wishes to do so for particular events.

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

Declared In
NSTextAttachment.h

Document Revision History

This table describes the changes to *NSTextAttachmentCell Protocol Reference*.

Date	Notes
2006-05-23	First publication of this content as a separate document.

Document Revision History

Index